ESCOLA
POLITÉCNICA DA
UNIVERSIDADE DE
SÃO PAULO

DEPARTAMENTO DE
ENGENHARIA DE
ELETRICIDADE

LABORATÓRIO DE
SISTEMAS DIGITAIS

DISSERTAÇÃO DE MESTRADO

# PROJETO LÓGICO DA UNIDADE DE CONTROLE DE UM MINICOMPUTADOR

EDSON FREGNI

ORIENTADOR: Prof. GLEN GEORGE LANGDON, JR.

1972

FD-910

# PROJETO LÓGICO DA UNIDADE DE CONTROLE DE UM MINICOMPUTADOR

EDSON FREGNI

DISSERTAÇÃO DE MESTRADO
APRESENTADA À ESCOLA POLITÉCNICA
DA UNIVERSIDADE DE SÃO PAULO
- 1972 -

orientador:

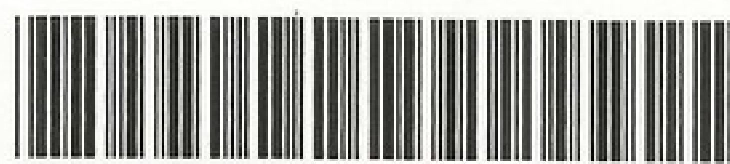
Prof. GLEN GEORGE LANGDON, Jr.

Dedicado a

Sr. Segundo e D. Recordina, meus pais,
a quem eu tento imitar.

Gilda, minha esposa, cujo apoio é essencial
para o meu trabalho.

DEDALUS - Acervo - EPEL



31500012274

# AGRADECIMENTOS

Gostaria de tornar público meu reconhecimento a:

Prof. Antonio Hélio Guerra Vieira, Prof. Oswaldo Fadigas Fontes Torres e Prof. Luiz de Queiroz Orsini pelo constante apoio e orientação;

Prof. Glen George Langdon, Jr., Eng. Célio Yoshiyuki Ikeda e engenheirando Flávio Celidônio Meirelles pela efetiva e decisiva participação em todas as fases deste trabalho.

Engenheirandos Flávio Celidônio Meirelles, Stephan Kovach e Edit Grassiani pela colaboração durante o projeto lógico e preparação deste texto;

Eng. Selma Shin Shimizu, Eng. Laércio Antonio Marzagão, Eng. Nestor de Mattos Cunha Jr. e técnicos Giácomo Henrique D'oro e Luiz Carlos Viegas pela implementação física;

Engenheirandos Aldo José Kuhl Jr., Sérgio Gonçalves, Marcelo Pessoa, Wilson Vicente Ruggiero, José Antonio Telles Guerra e Jorge Ramón D'Acosta Rivera pela colaboração em várias etapas do trabalho;

Eng. Antonio Marcos Aguirra Massola, Eng. Victor Mammana de Barros e engenheirando Haroldo Guibu pela preparação dos programas que constam neste trabalho;

Eng. Antonio Marcos Aguirra Massola, Eng. João José Neto, Eng. Benício José de Souza e engenheirando Luiz Henrique Tadeu Pedroso, que com a experiência de preparação do "software" colaboraram com valiosas sugestões de modificação do "hardware";

Eng. Victor Mammana de Barros, Prof. James Gregory Rudolph e Prof. José Rubens Dória Porto pelas discussões de inúmeros tópicos deste trabalho;

Sra. Gilda Fregni, minha esposa, pela revisão e datilografia;

Sra. Neide de Carvalho Pesso pela ajuda na pesquisa bibliográfica;

Sr. Ademir Souza Vianna, Sr. Miguel Caper e Sr. Paulo Monteiro pela impressão deste texto.

É claro que todos os professores, engenheiros, estagiários e técnicos do Departamento de Engenharia de Eletricidade da Escola Politécnica da USP, que participaram do projeto do computador referido neste texto, direta ou indiretamente colaboraram com o projeto da unidade de controle, por isso, meu agradecimento é extensivo a todos.

São Paulo, agosto de 1972

EDSON FREGNI

# SINOPSE

Após breve descrição da organização dos minicomputadores, fez-se uma análise geral das unidades de controle, dando-se ênfase à classificação: unidade de controle fixa e unidade de controle microprogramada. A seguir descreveu-se a arquitetura e o fluxo de dados do minicomputador do LSD. Então, detalhou-se o projeto lógico da unidade de controle desse minicomputador e terminou-se com a análise das falhas detetadas durante a fase de depuração do protótipo. No apêndice propôs-se algumas alterações no sistema.

# ABSTRACT

After a brief discussion of minicomputer organization, a general analysis of control units is presented , emphasizing the hardwired and microprogrammed control unit classification. Then a description is given of the LSD minicomputer architecture and data flow. After that the logic design of that minicomputer control unit is discused in detail. This is followed by an analysis of the faults detected in the prototype, debug phase. In the appendix some new proposals are made for changes in the system.

# CONTEÚDO

1- INTRODUÇÃO .................................... PG 1

2- COMPUTADORES DIGITAIS

2.1 - Histórico ............................ PG 2
2.2 - Os Minicomputadores .................. PG 3
2.3 - Estrutura dos Minicomputadores ......... PG 3

3- UNIDADES DE CONTROLE

3.1 - Conceitos Gerais ...................... PG 5
3.2 - Controle Fixo ......................... PG 7
3.3 - Unidade de Controle Microprogramada .... PG 9
3.4 - Comparação entre as duas Técnicas de controle ........................... PG 12

4- O MINICOMPUTADOR DO LSD

4.1 - Introdução ............................ PG 15
4.2 - Descrição da Arquitetura .............. PG 15
4.3 - Descrição do Fluxo de Dados ............ PG 23
4.4 - Outros Detalhes ....................... PG 31

5- PROJETO LÓGICO DA UNIDADE DE CONTROLE

5.1 - Introdução ............................ PG 36
5.2 - Determinação do Ciclo da Máquina e projeto lógico do Relógio Central ........ PG 39
5.3 - Projeto do Decodificador .............. PG 40
5.4 - Projeto do Controle de Estado .......... PG 41
5.5 - Definições do Projeto do Gerador dos Sinais de Controle ................... PG 45
5.6 - Determinação da Carta de Microoperações PG 46
5.7 - Árvore dos Sinais de Controle e Expressões Booleanas ......................... PG 50
5.8 - Testes ................................ PG 52
5.9 - Síntese dos circuitos ................. PG 53
5.10- Partição .............................. PG 54
5.11- Documentação .......................... PG 56
5.12- Sumário ............................... PG 57

6- OBSERVAÇÕES FINAIS

6.1 - Depuração do Sistema .................. PG 58
6.2 - Análise dos Erros ..................... PG 59
6.3 - Conclusões ............................ PG 59

7- APÊNDICE

7.1 - Restauração Automática de V e T ....... PG 61
7.2 - Nova Instrução: Soma $V_A$ no Acumulador .. PG 62
7.3 - Índice Negativo ....................... PG 62
7.4 - Endereçamento Indireto ................ PG 63
7.5 - Estatística das Instruções ............ PG 64

# 1- INTRODUÇÃO

O objetivo primeiro desta dissertação é fornecer uma descrição detalhada da técnica usada no projeto da unidade de controle do computador projetado e montado por uma equipe do Laboratório de Sistemas Digitais do Departamento de Engenharia de Eletricidade da Escola Politécnica da Universidade de São Paulo.

A escassez de matéria escrita nessa área é justificada pela falta de sistematização com que são conduzidos projetos dessa natureza. O projeto da unidade de controle do tipo controle fixo exige muito do projetista, e seria, segundo muitos autores, uma arte. Procurou-se dar sistematização ao projeto, desviando-se um pouco do ótimo para assegurar-se uma taixa de erros baixa. Sempre que possível recorreu-se à ajuda do computador para diminuir as falhas humanas. Deu-se muita importância à documentação farta e eficiente durante todas as etapas, já que o projeto da unidade de controle era diretamente relacionado ao projeto do fluxo-de-dados.

O segundo propósito desta dissertação é descrever didàtica e organizadamente o minicomputador para que sirva como orientação dos projetos futuros.

O terceiro objetivo é documentar, com alguns detalhes a unidade central e principalmente a unidade de controle , desse minicomputador.

Tendo em vista esses objetivos, os capítulos 2 e 3 têm por finalidade situar o leitor dentro dos temas: minicomputador e unidade de controle - ali, procurou-se expor os assuntos de uma forma geral, sem detalhes, fornecendo ao interessado uma farta referência bibliográfica. O capítulo 4 descreve o minicomputador do LSD. O capítulo 5 é a descrição propriamente dita do trabalho de mestrado - é ali onde se encontra a parte central deste trabalho. Para o capítulo 6 reservou-se as conclusões e observações finais. Como apêndice anexou-se os detalhes do projeto, como circuitos, listagens, etc., que têm por objetivo a documentação já citada.

# 2- COMPUTADORES DIGITAIS

## 2.1: HISTÓRICO

Em 1671 quando Leibniz, ao aperfeiçoar uma calculadora mecânica inventada por Blaise Pascal em 1642, afirmava que "... os astrônomos seguramente não terão que continuar com os pacientes exercícios que são requeridos pela computação..."[1], mostrava que, já naquela epoca, o trabalho mecânico, mesmo o mental, era relegado a uma categoria inferior.

A maioria dos autores concordam que a primeira idéia de uma máquina com a filosofia dos computadores digitais atuais, surgiu em 1833 quando o matemático inglês Charles Babbage concebeu o seu " Engenho Analítico ". Nievergelt afirma que "Babbage e vários contemporâneos provaram, por seus textos, um completo entendimento dos princípios básicos com os quais um computador para uso geral opera"[2].

Contudo foi necessário que se esperasse o desenvolvimento da tecnologia para que se pudesse por em prática a idéia de Babbage.

Em 1939 a IBM construiu o Harvard Mark I, um computador a relês projetado na Universidade de Harvard. No final da década dos 30 e início da de 40 Universidades e indústrias construiram vários computadores eletromecânicos.

Durante a segunda grande guerra, em 1943, a Moore School of Engineering desenvolveu, na Pennsylvania, o primeiro computador eletrônico: o ENIAC. Era o início da primeira geração dos computadores, construídos com válvulas. No início da década de 50 ainda se construia computadores da 1º geração ( IBM 702 pronto em 1954 )(*). Do ponto de vista do impacto econômico, os computadores nasceram com o ENIAC, daí porque é chamada de 1º geração.

Em 1948 a Bell Telephone Labs inventou o transistor e em 1954 iniciou os testes do TRADIC (Transistor Digital Computer), um computador construído com 700 transistores. Era a segunda geração nascendo, caracterizada por essa tecnologia, que se extenderia até a década de 60.

Os circuitos integrados surgiram em 1960, e a seguir essa tecnologia foi totalmente absorvida pelos computadores e estava então caracterizada a terceira geração.

Comparando-se essas máquinas, pode-se dizer que de geração a geração, a velocidade aumenta significativamente e sua complexidade cresce em ordens de grandeza (ENIAC tinha 18 000 válvulas, o IBM STRETCH[3] em 1961 incorporava 150 000 transistores, e o ILLIAC IV[4] tem circuitos integrados equivalentes a vários milhões de transistores). Além de outras evoluções em "hardware" como confiabilidade, tamanho físico, potência consumida, há significativas diferenças em "soffware". Este, antes considerado problema de escala menor é, atualmente a principal preocupação dos fabricantes de sistemas de grande porte.

Langdon[5] apresenta um estudo completo da evolução dos computadores principalmente em termos de "hardware", enquanto que Rosin[6] se detém na evolução do "software".

---

(*) Em sua época, o IBM 702 era conhecido como segunda geração. Mas atualmente ele é classificado como fim da primeira geração ("late first generation").

## 2.2 OS MINICOMPUTADORES

Distingue-se facilmente um minicomputador de um sistema de grande porte pelo seu baixo custo, pequena capacidade e tamanho de palavras reduzido.

Pode-se dizer que o PDP-8 da "Digital Equipment Corporation" foi o primeiro minicomputador da 3ª geração de grande sucesso no mercado americano. É um computador de palavra de 12 bits e com memória de 4K palavras(7). De uma relação de "minis" publicada por Bhushan(8) pode-se notar que os minicomputadores são caracterizados pelos seguintes parâmetros:

palavra - 8 a 18 bits
memória - 1k a 32 k palavras
ciclo da máquina - 0,8 a 2.2μseg

Os minicomputadores são ideais para aplicações em tempo real por causa do seu baixo custo e versatilidade de entrada e saída. Bhushan(8) afirma que "a facilidade com que outros componentes do sistema podem ser ligados ao minicomputador é uma de suas mais importantes características".

Outra característica importante do minicomputador é sua aplicação: Morris (9), em um estudo comparativo, afirma que "talvez a característica mais comum à maioria dos minicomputadores é que êles são inteiramente dedicados a uma tarefa específica, enquanto que os sistemas de grande porte são freqüentemente usados em uma grande variedade de propósitos".

Pode-se afirmar que a maioria dos minicomputadores são de baixo custo, dimensões pequenas, baixa capacidade e destinados a uso específico. Contudo existem muitos "minis" de uso geral, apenas restritos pela sua capacidade física. Neste caso eles perdem um pouco da potencialidade que teriam se fossem projetados para fins determinados.

## 2.3 ESTRUTURA DOS MINICOMPUTADORES

Os minicomputadores têm estrutura muito parecida à dos computadores de primeira geração. Em geral têm instruções de endereço simples com acumuladores como operandos implicados; exploram as filosofias de endereçamento indireto, relativo e abreviado dadas as pequenas dimensões das palavras; operam com dados codificados em ponto fixo para economia de "hardware".

Na figura 2.1 vê-se as unidades básicas de um minicomputador característico. As linhas cheias sugerem caminhos de dados e as pontilhadas, sinais de controle.

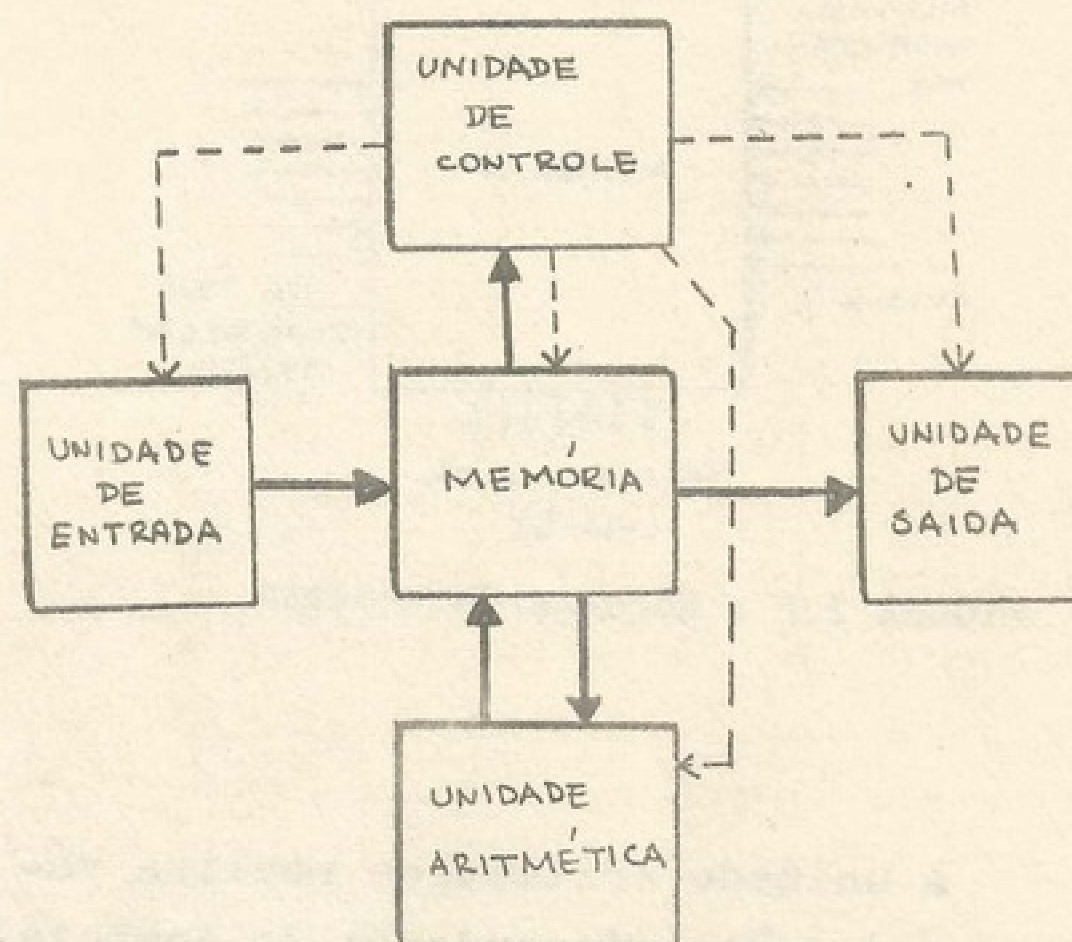



FIGURA 2.1 : Partes principais de um minicomputador

Esse é um esquema de um computador do tipo "programa-armazenado-na-memória de-dados", idéia proposta por J. von Neumann em 1947. Ali, um programa é armazenado com instruções colocadas em posições sucessivas da memória, que a unidade de controle vai retirando uma a uma e coordenando a sua execução.

### 2.3.1 MEMÓRIA(10)

Pode-se entender memória como mostrada na figura 2.2. Nesse esquema, um sinal na linha "escrever", inicia a gravação da palavra da "via E" na posição de memória endereçada pela via ("bus") de endereçamento; e com um sinal na linha "ler" o conteúdo da posição da memó

ria endereçada pela via de endereçamento é colocada na via S de saída.

2.3.2 UNIDADE ARITMÉTICA

Unidade aritmética é um circuito, conforme o esquema da figura 2.3, que opera com o conteúdo do acumulador e o operando que vem da memória, e o resultado é colocado no acumulador. O acumulador é um registrador, constituído de flip flops, que armazena uma palavra completa. O operando, lido na memória, é aquele endereçado pela instrução.

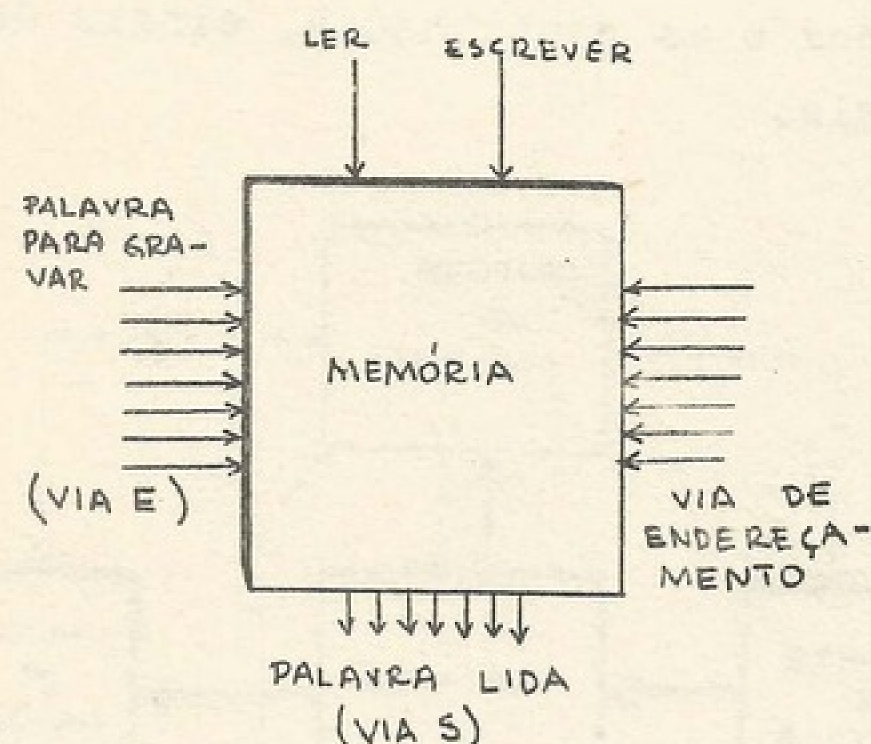



FIGURA 2.2 : Esquema da memória

A unidade aritmética realiza várias operações (dependendo do computador) como soma, subtração, multiplicação, funções booleanas (AND, OR, XOR, etc.), comparação, etc. Por isso existe uma terceira entrada no circuito: aquela que especifica o tipo de função da Unidade Aritmética.

2.3.3 ENTRADA/SAÍDA[10]

Em geral, os minicomputadores realizam a entrada e saída de dados diretamente da memória com uma técnica chamada "acesso-direto-à-memória" (direct memory acess) onde a unidade de entrada e saída "para" o relógio central e "rouba" (cycle-steal) um ciclo da unidade de controle para ler ou armazenar dados diretamente na memória.

Com o relógio central parado, a unidade central não continua o processamento normal do programa já que ela não recebe os sinais de referências de tempo. Enquanto isso, se for uma operação de saída, a entrada/saída comanda um ciclo de leitura da memória para retirar a palavra que deve ser enviada para o dispositivo de saída. Se for uma operação de entrada, a palavra que chega do dispositivo é gravada na memória durante esse ciclo roubado.

Os minicomputadores exploram bastante a técnica de interrupção[11], na qual um dispositivo de entrada e saída envia um sinal elétrico para a unidade de controle que então, interrompe o seu trabalho normal (programa principal) para executar uma sub-rotina de atendimento desse dispositivo, e voltar para o trabalho inicial.

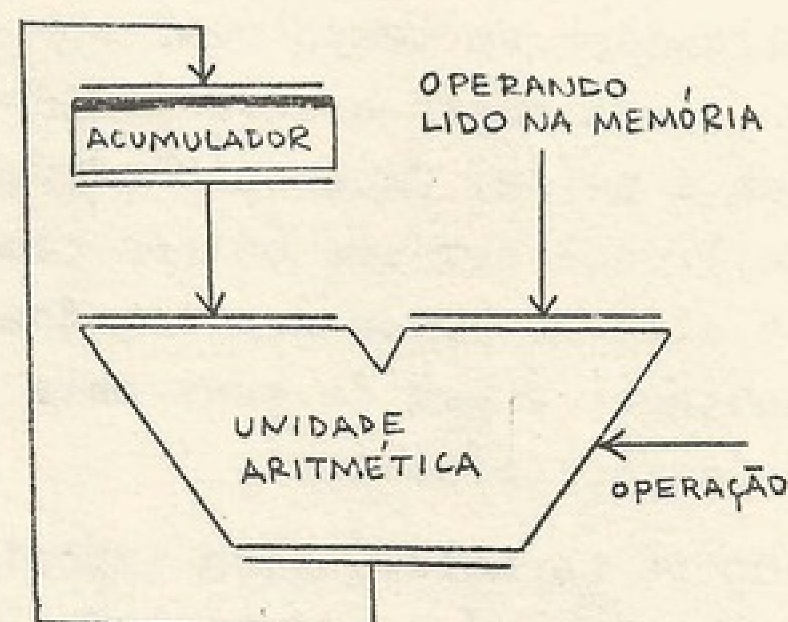



FIGURA 2.3 : Esquema da unidade aritmética de um minicomputador

# 3-UNIDADES DE CONTROLE

## 3.1 CONCEITOS GERAIS

Langdon[10] diz que "o fluxo-de-dados (*) é como uma marionete, com os fios e as linhas de controle disponíveis para quem quiser manobrá-la". É função da unidade de controle atuar sobre essas linhas de modo a gerenciar a execução de um programa em todos os detalhes. O fluxo de dados é uma ferramenta usada pela unidade de controle.

A figura 3.1 é um esquema mostrando as entradas e saídas da unidade de controle. Sua função é, conhecendo a instrução que está sendo executada, gerar todos os sinais para que essa instrução seja executada no fluxo de dados e, dando prosseguimento ao processo, providenciar a retirada da próxima instrução da memória. O operador do sistema pode alterar o funcionamento normal do sistema através das chaves do painel.

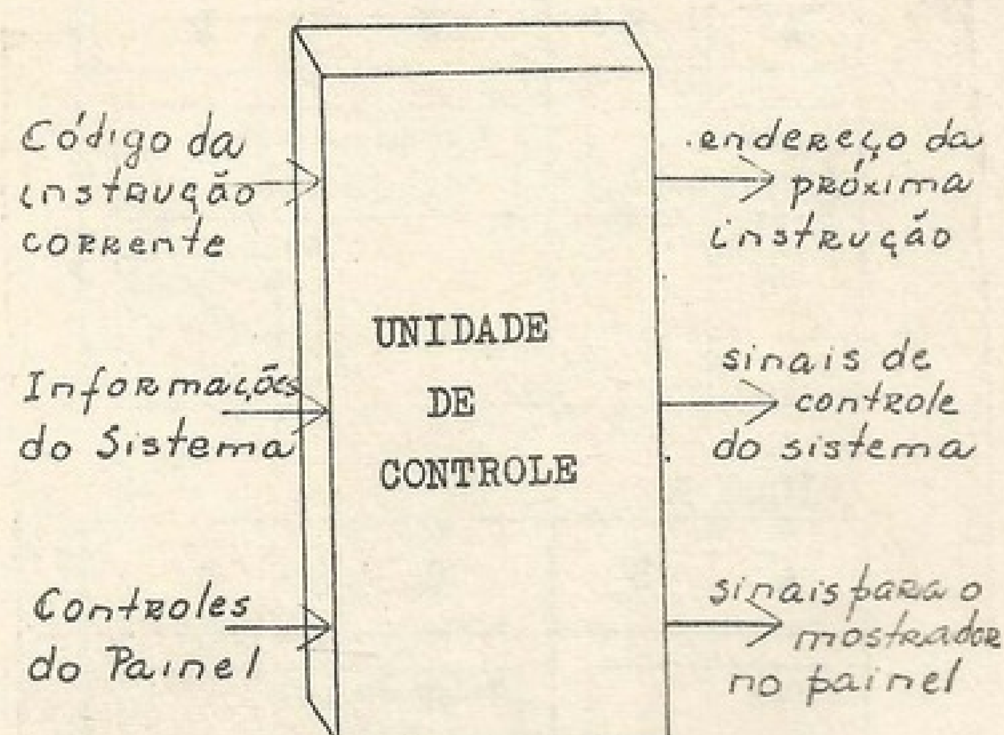



**Fig. 3.1 - Esquema da unidade de controle**

Uma instrução é levada a cabo com a execução de uma sequência de operações elementares características da máquina. Dá-se o nome de microoperações[12] a essas operações elementares. Para exemplificar, pode-se dizer que são microoperações: transferência de dados de um registrador para outro (A→C), limpar (tornar zero) algum flip flop (0→B), somar "um" ao conteúdo de um registrador (CI←CI+1), ler memória, etc.

Uma microoperação é concluída com a conjunção ou sequência de inúmeros sinais elétricos, corretamente ordenados. Por exemplo, para se executar a microoperação somar "um" ao conteúdo de um registrador, precisa-se de sinais elétricos que: coloquem o conteúdo desse registrador em uma das entradas do somador, coloquem o número "um" na outra e a saída do somador na entrada do registrador e, depois de algum tempo, copiem a saída do somador no registrador.

Uma das responsabilidades da unidade de controle é gerar os sinais elétricos de modo a realizar a microoperação, e com o conjunto de microoperações na ordem certa, executar a instrução. Além disso, a unidade de controle deve estar pronta a responder a qualquer comando do painel ou de qualquer dispositivo de entrada e saída, iniciando, desviando ou parando o seu trabalho.

A unidade de controle, em seu trabalho, toma como referência de tempo um intervalo chamado ciclo da máquina, que é usualmente estabelecido com referência ao ciclo de memória[13]. Usualmente, divide-se o ciclo da máquina em vários segmentos e cada microoperação dura um ou mais segmentos. Esses segmentos são a menor unidade de tempo do sistema, ou seja dois eventos não simultâneos distam de um múltiplo desses segmentos. Na figura 3.2 ve-se um exemplo de ciclo de máquina dividido em segmentos.

---

(*) Muitos autores dividem a unidade central de processamento (UCP) em duas partes : fluxo-de-dados ("data flow") e unidade de controle. O fluxo-de-dados é a parte que inclui os caminhos e paradas dos dados dentro do sistema, como a memória, registradores centrais, unidade aritmética, vias ("busses") de entrada e saída, etc.

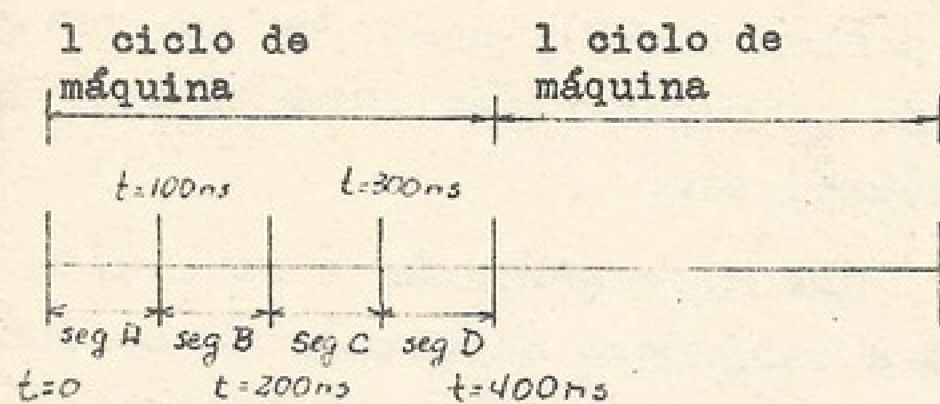



**Fig. 3.2 - Exemplo de um ciclo de máquina dividido em segmentos de tempo**

O conceito de ciclo de máquina varia de situação a situação e de autor para autor.

A execução de uma instrução de endereço simples gasta, sistematicamente, dois ciclos de máquina: o primeiro deles, que se usa para ler essa instrução da memória, é chamado de CICLO-I (ciclo de instrução ou de "fetch"), e o segundo ciclo, durante o qual lê-se o operando da memória e realiza-se a operação especificada, é chamado de CICLO-E (ciclo de execução). É claro que existem instruções com vários ciclos de execução (como a multiplicação em um sistema que tenha apenas somadores) e aquelas com nenhum ciclo de execução, como as instruções de desvio que servem apenas para endereçar a próxima instrução.

Por exemplo, seja um computador com um acumulador que deve executar a instrução "soma no acumulador", onde se especifica um endereço cujo conteúdo é o operando que se deve somar. Suponha-se ainda que a memória seja magnética, portanto com leitura destrutiva(10), e necessita-se restaurar o que se lê. A figura 3.3 mostra um esquema das microoperações geradas, usando um bloco de atraso esquemático. Imagine-se um pulso entrando em E e à medida que vai passando, vai gerando as microoperações no tempo certo. Gschwind(14) usa esse tipo de gráfico ilustrativo.

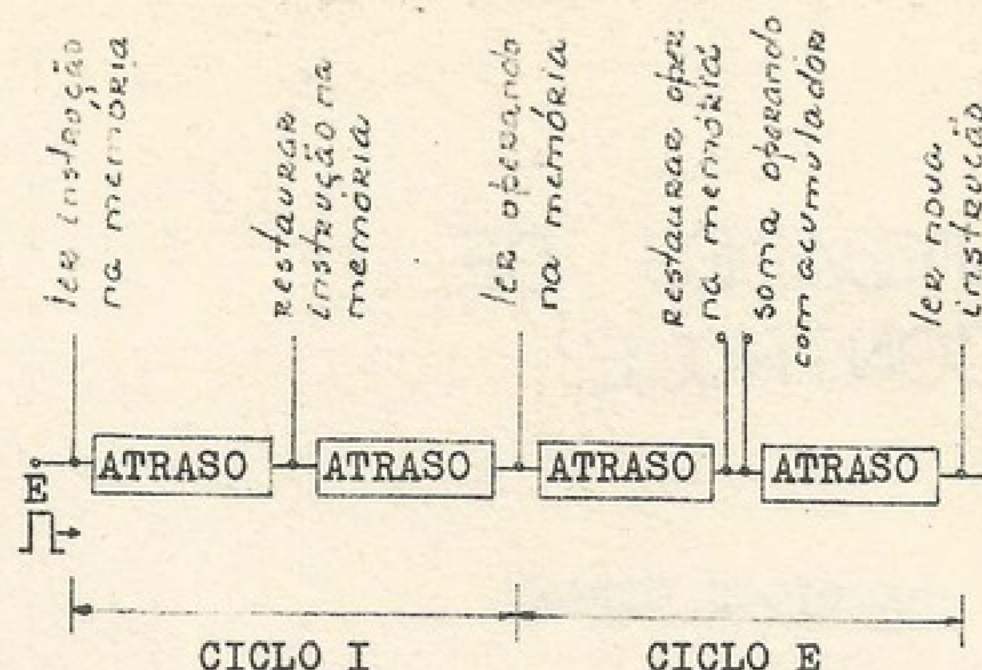



**Fig. 3.3 - diagrama esquemático das microoperações numa instrução de soma.**

Note-se que o ciclo I é o mesmo para todas as instruções, havendo ramificação no gráfico (fig.3.3) apenas no ciclo E, depois da decodificação do código da instrução.

Uma maneira muito usada atualmente para se descrever a sequência de microoperações na execução de uma instrução são as chamadas cartas de micro-operações ("timing charts"(15)) onde se fornece um diagrama cuja referência são os ciclos I e E com seus segmentos de tempo. A figura 3.4 mostra a carta de microoperação do exemplo anterior.

CICLO I

| A | B | C | D |
|---|---|---|---|
| ler memória | | restaurar memória | |
| | | | endereçar operando |

CICLO E

| A | B | C | D |
|---|---|---|---|
| ler memória | | restaurar memória | |
| | | AC + OPER → AC | endereça próxima instr. |

**Fig. 3.4 - Carta das microoperações da instrução de soma do exemplo**

Outra função da unidade de controle é zelar pelo estado do sistema Se qualquer situação anormal ocorrer

(erros de paridade, dispositivo de entrada e saída desligados, etc) ela deve parar a execução, esperando a atuação do operador. Pode-se classificar os controles do painel, em dois grupos: aqueles que servem para carregar ou agir sôbre um programa (partida, pare, chaves de dados, carrega endereço, etc) e aqueles que servem para depurar ("debug") um programa (mostra posição, ciclo único, instrução única, etc).

Resumindo, pode-se dizer que Unidade de Controle é a parte do sistema que retira as instruções da memória, seguindo a sequência do programa, e as executa através da combinação das microoperações.

Possíveis classificações das Unidades de Controle

SIEGEL[16] propõe a classificação das unidades de controle em dois grupos, síncrono e assíncrono, dizendo que na máquina síncrona o gerador dos sinais de segmento de tempo ("timing-pulse generator") produz um conjunto de sinais de controle em cada ciclo e as microoperações são geradas tomando apenas esses sinais como referência, enquanto que na máquina assíncrona, o fim de uma microoperação inicia a seguinte. Contudo ele termina dizendo que muito poucas máquinas são inteiramente síncronas ou assíncronas, a maioria usa uma combinação de ambas as técnicas.

Existe ainda o conceito de unidade de controle centralizada e descentralizada, apresentado por EADIE[17]. Um sistema de controle centralizado é aquele onde existe uma única seção do sistema que gera todos os sinais de tempo, de controle e de "clock". É uma organização normalmente usada em minicomputadores, que tem a vantagem de ser direta, positiva e de controle pouco complexo. Nos sistemas de controle descentralizado, a unidade de controle apenas monitora e dirige as várias subseções do computador. Esse método tem a vantagem de que as várias porções do sistema podem operar separadamente em sua própria velocidade ótima.

Quanto à sua implementação, classifica-se as unidades de controle em dois grupos: as de controle fixo (através de circuitos, "hardware") e as micro programadas. Essa classificação é importante já que leva em conta a implementação física e portanto depende da tecnologia. Por isso os itens seguintes detalham o assunto.

## 3.2 CONTROLE FIXO

Em um computador de controle fixo, a unidade de controle é um circuito ("hard-wired") encarregado da geração dos sinais de controle, a partir do código da instrução corrente. Projeta-se esse circuito para uma dada arquitetura, e quando se quiser modificá-la será necessário reprojetar a unidade de controle. William Roberts[18] afirma que "se a unidade de controle é fixada por circuitos (hard-wired), o computador funciona melhor em apenas uma das classes de aplicações e menos eficientemente em todas as outras". Outro conceito é o da unidade de controle microprogramada, na qual a unidade de controle não é fixada por circuitos e é absolutamente geral, com seu comportamento ditado por subrotinas armazenados na sua memória de controle.

A figura 3.5 mostra, esquematicamente as partes principais de uma unidade de controle do tipo controle fixo. ROBERTS[18] apresenta um esquema parecido com esse, porém com menos detalhes. Segue a análise de cada bloco separadamente.

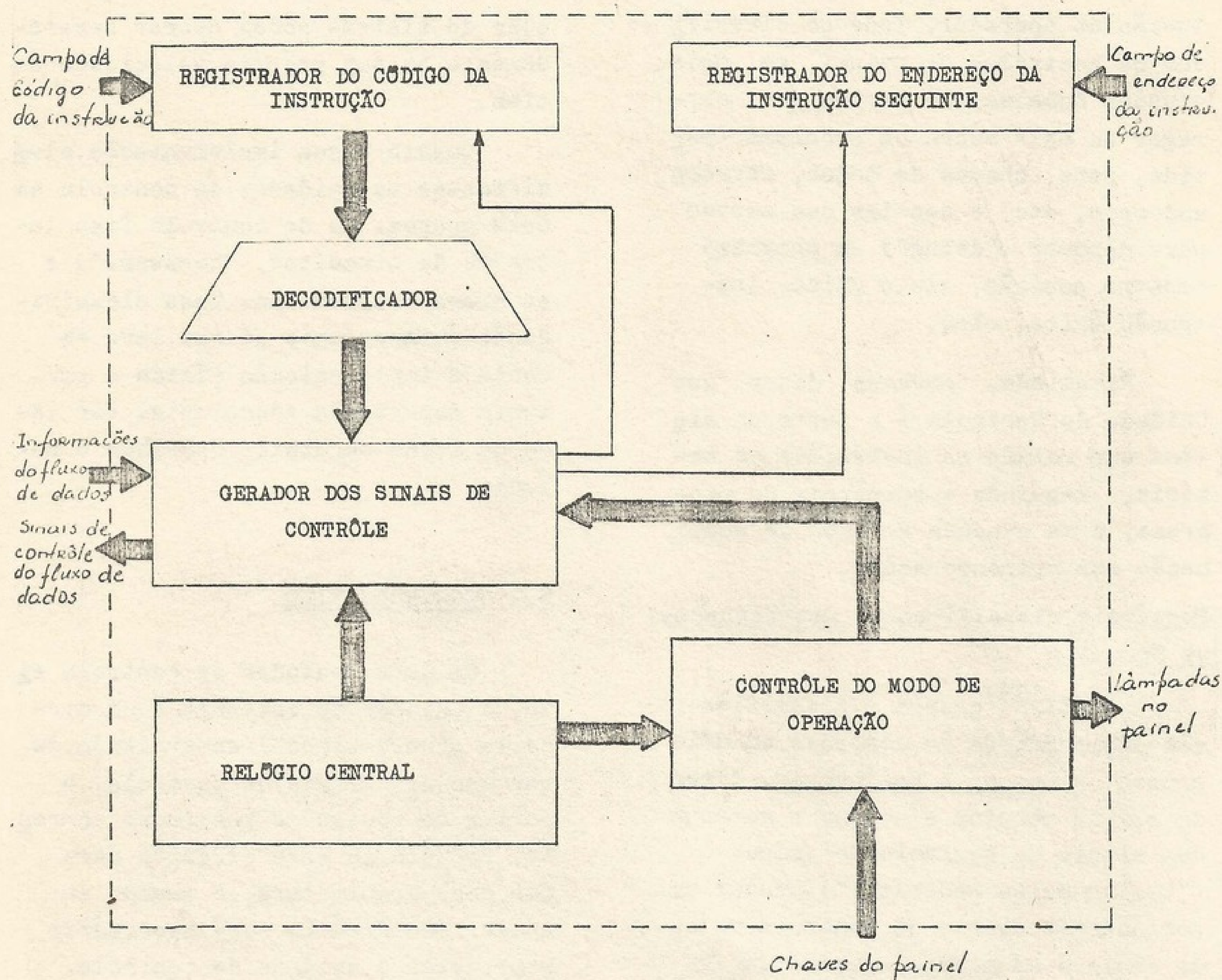



**Fig. 3.5 - Esquema de uma unidade de contrôle do tipo contrôle fixo**

<u>Registrador do código da instrução</u>

Sua função é apenas de guardar o código da instrução, para alimentar o decodificador durante o tempo de execução da instrução.

<u>Registrador de endereço de instrução</u>

Muitas vezes chamado de contador de instruções ("instruction counter[19]") ou contador de programa ("program counter[18]), tem a finalidade de endereçar a instrução seguinte. Em instruções sem desvios ( pulos - ver capítulo seguinte), ele é incrementado de <u>um</u>, e no caso de desvios é carregado com o campo de endereço da instrução.

<u>Decodificador</u>[10,11]

É um circuito convencional, cujas saídas são as instruções. Existem tantas linhas de saída quantas forem as instruções, e sempre apenas uma das saídas fica no nível "um", indicando a instrução corrente.

Gschwind[14] chama a esse bloco de <u>tradutor de função</u> (" Function Translator"), e o que é importente para os capítulos seguintes, afirma que "o decodificador pode também fornecer saídas especiais que são energizadas para certos grupos de instruções, tais como instruções de testes, instruções de entrada/saída, etc".

Gerador dos sinais de controle

Tomando como referência os sinais de tempo (ciclo I ou ciclo E fornecido pelo controle de modo de operação e sinais de segmentos de tempo fornecidos pelo relógio central) e a instrução corrente, o circuito gerador de sinais de controle, gera os sinais que irão controlar o fluxo de dados. Projeta-se essa parte da unidade de controle a partir da carta de microoperações. Ware(19) chama essa parte de sequencializador (" sequencing") e diz que os sinais gerados "são distribuídos nos lugares certos nas ocasiões corretas através de portas que são abertas pelo decodificador.

Relógio Central

É um circuito com um oscilador básico que gera sinais que definem segmentos de tempo conforme visto na figura 3.6. É a referência de tempo

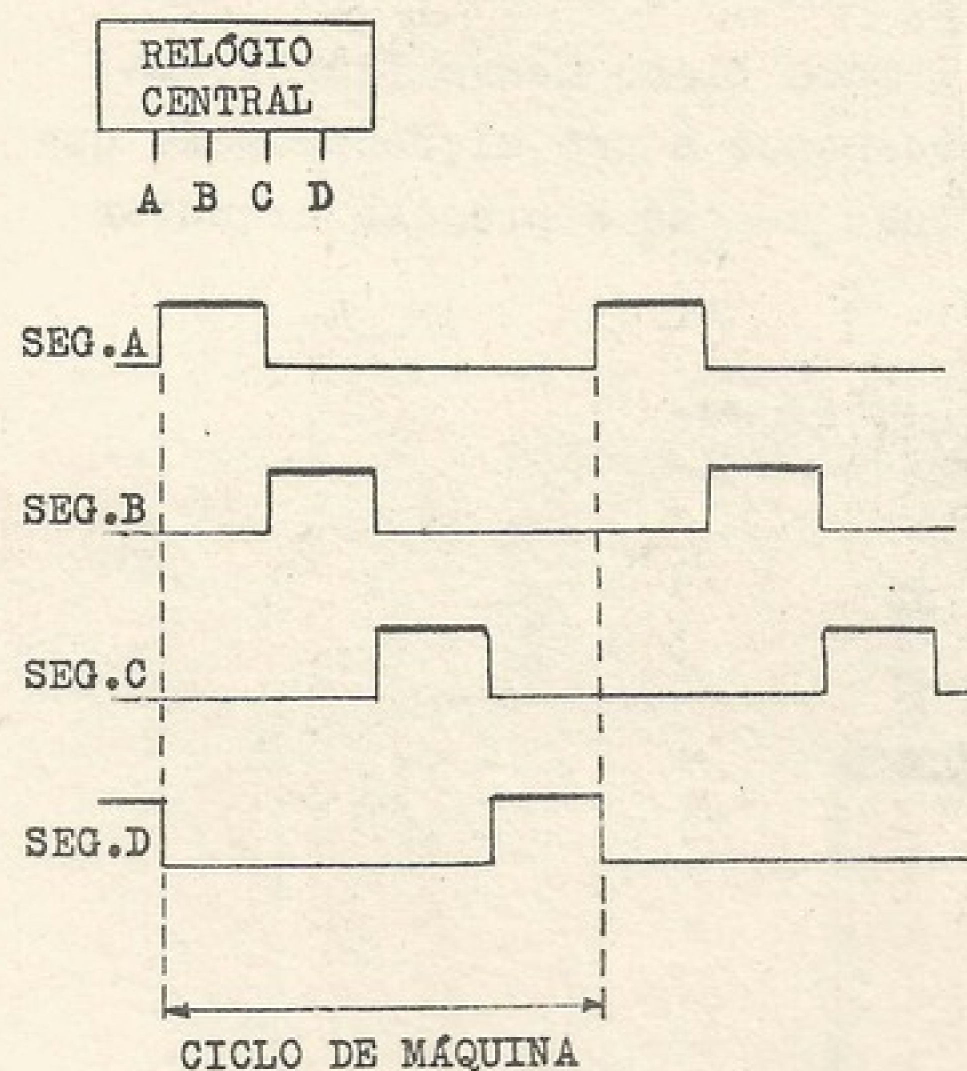



**Fig. 3.6 - Sinais gerados por um relógio central com quatro segmentos de tempo**

para o sistema. Por exemplo, com o sinal A controla-se os eventos do início do ciclo, com o B os do final da primeira metade e assim por diante. Portanto, pode-se dizer, que o relógio central especifica uma sequência fixa de posições no tempo dentro do ciclo da máquina.

Controle do modo de operação

É a parte de interface entre o sistema e o operador. Por isso, o funcionamento da unidade de controle depende totalmente de sinais enviados por esse bloco. Quando, durante o funcionamento, o operador aciona a chave de PARE ("stop"), esse circuito espera que se termine a instrução corrente e para o sistema. É ele também que decide quando é ciclo de instrução ou quando é de execução. Interrupções provocadas por dispositivos de entrada e saída são atendidas por ele.

Esse bloco recebe o nome de controle do modo de operação pois os modos especiais de funcionamento como instrução única ou ciclo único, são executados por ele.

Resumindo

Pode-se dizer que controle fixo é um circuito projetado para "manobrar" o fluxo de dados na sequência correta das microoperações. E que recebeu o nome de fixo pois não pode ser mudado facilmente, exigindo que seja reprojetado no caso de alguma alteração.

## 3.3 UNIDADE DE CONTROLE MICROPROGRAMADA

Em 1951 o matemático inglês M. V. Wilkes do Laboratório Matemático da Universidade de Cambridge apresentou a primeira idéia de microprogramação.

Por isso para introduzir esse assunto nada melhor que usar as próprias palavras de Wilkes(32)

"Em muitas máquinas o projeto da Unidade de Controle tem sido obtido através de métodos semi-empíricos, e os circuitos resultantes, apesar de funcionarem são complexos e não sistemáticos, e poderiam exigir alterações consideráveis se alguma modificação apreciável precisasse ser feita no código

de operação da máquina. A finalidade de desenvolver um projeto simples em sua estrutura global e aplicável qualquer que seja o código de operação da máquina. Os princípios básicos foram primeiramente descritos, de uma forma breve, em uma conferência na Universidade de Manchester em 1951 é apresentado com maiores detalhes em um artigo publicado em 1953 ( M.V. Wilkes e J.B. Stringer, "MICROPROGRAMMING AND THE DESIGN OF THE CONTROL CIRCUITS IN AN ELECTRONIC DIGITAL COMPUTER", Proceedings of the Cambridge Philosophical Society, 1953, 49, part 2, pp. 230)."

Na figura 3.7 vê-se um esquema simplificado de uma unidade de controle microprogramada. Nessa estrutura usa-se o código de uma instrução, que não mais é decodificado, como endereço de uma "microsubrotina" (subrotina na memória de controle). Portanto para cada instrução tem-se uma "microsubrotina" armazenada na memória de controle, que é lida sequencialmente, microinstrução a microinstrução, e vai ditando todas as microoperações necessárias para a execução da instrução indicada pelo programador. A vantagem fundamental dessa técnica, é a versatilidade. Pode-se incluir no sistema instruções extremamente complexas, apenas mudando a microprogramação da unidade de controle.

Seja um exemplo: imagine-se, novamente, aquela instrução discutida no item 3.1 (soma do operando no acumulador). O ciclo I, também conhecido como fase de busca ("fetch"), corresponde a uma micro-subrotina na memória de controle em uma posição definida. Essa subrotina executa todos os ciclos I, se encarregando de ler a instrução na memória principal, extrair o código e endereçar a microsubrotina de execução dessa instrução na memória de controle. A figura 3.8 apresenta o fluxograma ("flow-chart") da microsubrotina da fase de busca do exemplo.

Cada bloco nesse fluxograma corresponde a uma microoperação que o fluxo de dados precisa executar

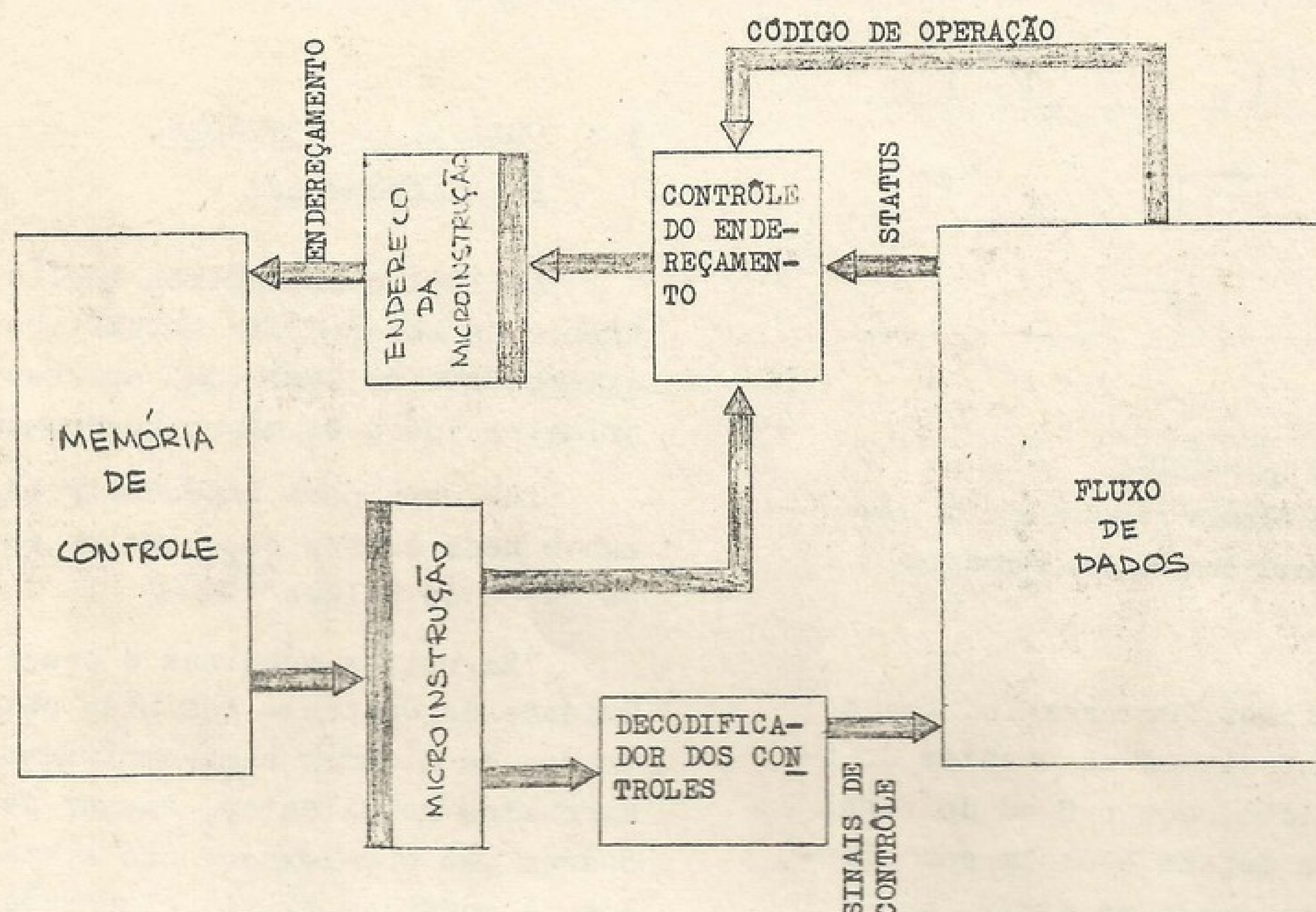



Fig. 3.7 - esquema simplificado de uma unidade de contrôle microprogramada

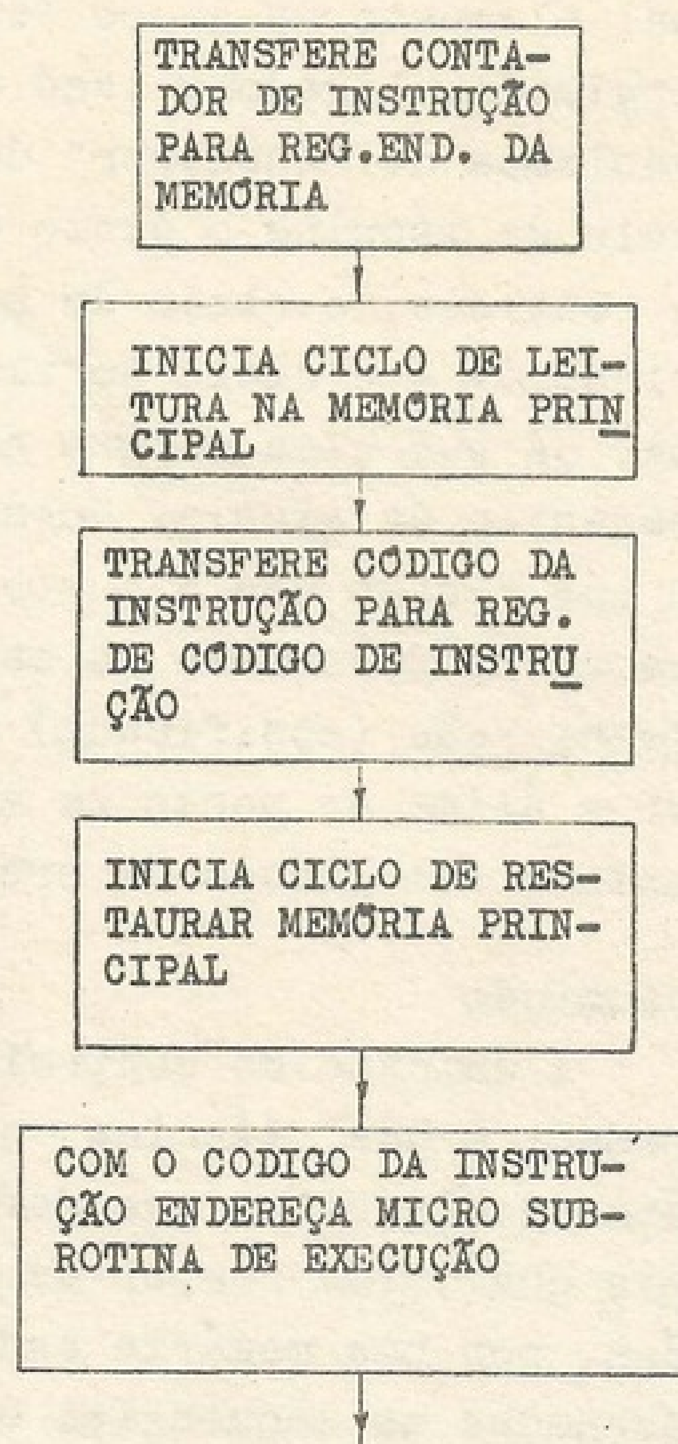



**Fig. 3.8 - Fluxograma do ciclo I num computador microprogramado**

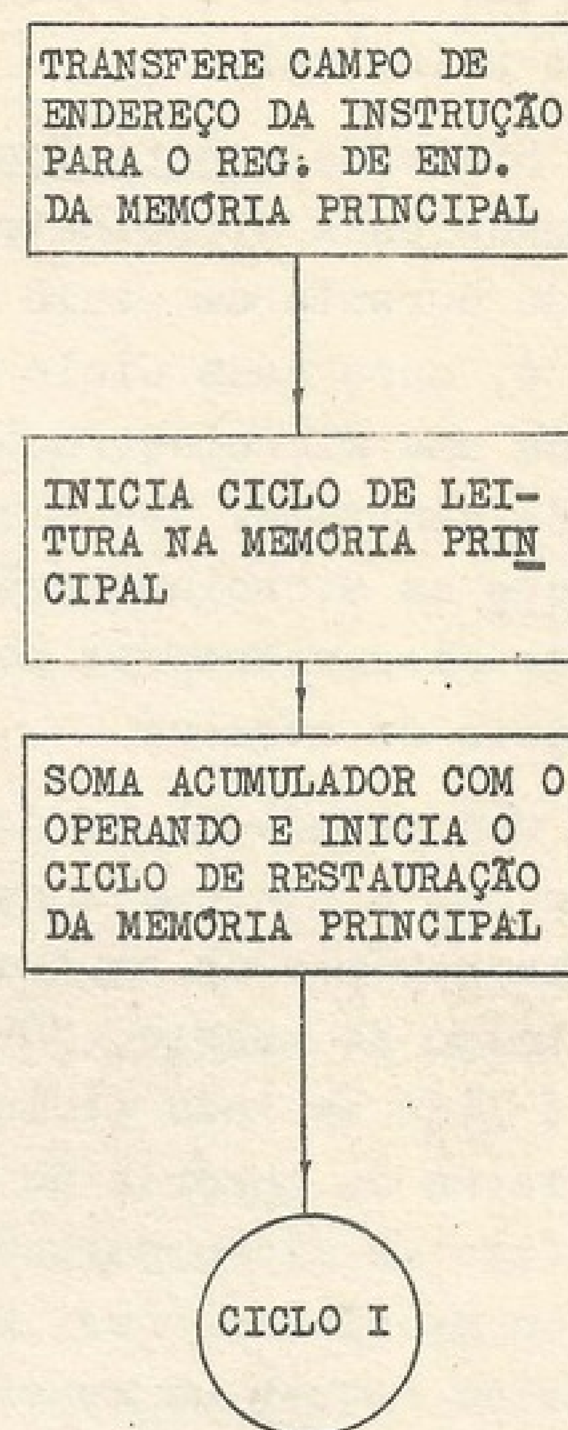



**Fig. 3.9 - Fluxograma da microsubrotina de execução de soma do exemplo**

e, no nosso exemplo, corresponde a uma microinstrução. Portanto a unidade de controle lê a primeira microinstrução (transferir CI para RE executa-a e parte para ler a seguinte (ler memória), idem e assim por diante. Terminado o ciclo I a microinstrução seguinte já pertence à microsubrotina de execução. mostrada na figura 3.9. Poder-se-ia dizer que existem tantas microsubrotinas quantas forem as instruções do sistema. Mas, é claro que, assim como na unidade de controle fixa o decodificador gera sinais comuns a grupos de instruções, existem também trechos de microsubrotinas comuns a várias instruções.

Note-se a total equivalência entre o fluxograma da microsubrotina e as cartas de microoperações apresentadas no controle fixo.

Dada a extensão do assunto microprogramação, procurou-se fornecer os elementos essenciais dessa técnica. Um estudo mais detalhado consta de livro de Husson[20].

Redfield[22] sugere três medidas que caracterizam a microprogramação: monofásicos/polifásicos, paralelo/série e codificado/não codificado.

<u>Monofásicos e Polifásicos</u>

É prática comum, ao se projetar um computador microprogramado organizar-se o fluxo de dados de tal modo que uma microoperação corresponda, na maioria das vezes, a uma volta completa dos dados através da unidade aritmética. Nesse caso, chama-se de <u>ciclo da máquina</u> (ás vezes de ciclo do fluxo de dados) ao tempo gasto nessa volta.

Assim, toda microoperação é executada em um ciclo de máquina, conceito este que, aqui, se desvincula da memória principal.

Sistemas monofásicos são aqueles onde cada microinstrução é executada durante um ciclo de máquina, isto é, para cada ciclo de máquina é lida uma microinstrução na memória de controle. Nos sistemas polifásicos as microinstruções especificam as microoperações para mais que um ciclo da máquina.

Comparando-se os dois pode-se dizer que o sistema monofásico exige uma memória de controle com menor tempo de acesso, já que a memória é lida em todo ciclo da máquina a palavra da memória de controle especifica as microoperações de um ciclo da máquina apenas. O sistema polifásico requer um menor número de microinstruções, já que cada microinstrução especifica a tarefa de vários ciclos da máquina.

Paralelo e série

É um conceito ligado ao tipo de endereçamento da memória de controle. No paralelo, o cálculo do endereço da microinstrução seguinte é feito simultaneamente à execução da microinstrução corrente. No série, é esperado o término da execução da microinstrução para se decidir sôbre a próxima. A segunda técnica, apesar de mais lenta, tem a vantagem de não oferecer problemas nos casos de desvios condicionais na microsubrotina, pois ali não se pode desviar em função do resultado da microoperação corrente.

Codificado e não codificado

Controle não codificado (horizontal[23]) é aquele onde cada ponto de controle do fluxo de dados corresponde a um bit da palavra de controle. No codificado, pontos de controle do fluxo de dados que nunca são energizados simultaneamente são agrupados e para cada grupo existe um campo codificado na palavra de controle que irá indicar qual elemento do grupo deverá ser energizado. Ligados a isso existem os problemas de "encaixar" dentro do ciclo da máquina o ciclo da memória de controle, o ciclo da memória principal e o ciclo do fluxo de dados, já que eles formam grupos independentes de sinais. Husson[20] afirma que "mesmo com o custo adicional dos decodificadores de campos, tal organização (codificada) mostrou ser a ótima do ponto de vista do custo e facilidade de programação.

Resumindo

A unidade de controle microprogramada é uma técnica que substituiu os circuitos que geravam os sinais que iriam compor as microoperações, por uma memória onde ficam armazenadas as sequencias de microoperações necessárias para executar cada operação. Vandling[24] apresenta o exemplo de uma unidade de controle microprogramada, com detalhes.

## 3.4 COMPARAÇÃO ENTRE AS DUAS TÉCNICAS DE CONTROLE

Seguem as vantagens da microprogramação citadas por inúmeros autores (18,22,24,25):

Facilidade de projeto: é inerente à técnica, dada a sua regularidade. Foi essa característica que levou ao seu desenvolvimento.

Flexibilidade: pode ser modificada facilmente, bastando alterar o microprograma na unidade de controle.

Manutenção: falhas são facilmente encontradas, provocada pela regularidade. São possíveis ainda microprogramas de diagnose [26,27] para ajudar na manutenção do sistema completo.

Velocidade: alguns autores[28] admitem que quando um usuário puder adaptar o seu repertório de instruções às suas necessidades próprias, ele conseguirá muito maior velocidade do sistema. O próprio Wilkes[33] mostrou-se receoso com essa técnica ao afirmar: "o uso de memória ler-escrever no lugar de ler somente para armazenar os microprogramas de tal forma que o programador possa montar seus próprios microprogramas foi mencionado como uma possibilidade interessante, mas eu duvido se um computador projetado dessa forma seja realmente necessário".

Emulação: a vantagem básica de um sistema microprogramado é ele poder emular[20,23] outras arquiteturas, isto é, um sistema microprogramado pode ter seu repertório de instruções adaptado de modo a simular totalmente uma outra arquitetura.

Mas, apesar de todas essas vantagens, deve-se ter em mente as palavras de Flynn[29] ao afirmar que "a diferença significativa entre máquinas de controle convencional e microprogramadas está na potência das instruções". Isto é, sistemas com instruções bastante complexas devem ser microprogramados, ao passo que aqueles cujas instruções são bastante simples, como a maioria dos minicomputadores (instruções de endereço simples, com segundo operando implicado), devem ter sua unidade de controle do tipo controle fixo. É, economicamente, uma melhor solução.

Husson[20] sugere que quanto maior for o sistema maior vantagem terá em ser microprogramado, e analisa as unidades de controle dos dois sistemas comerciais que pela primeira vez utilizaram a microprogramação: sistema IBM/360 e RCA SPECTRA 70. Quanto ao sistema IBM /360 modelo 50, Schnabel[30] não vacila ao concluir pelas vantagens tanto em custo quanto em performance da microprogramação. Mas esses sistemas são de grande porte. Redfield[22] analisando um sistema de porte médio, o UNIVAC'S C/SP concluiu pelas vantangens na performance da microprogramação já que o custo era o mesmo. Isso mostra que nada tem de estranho o fato da maioria dos minicomputadores terem controle fixo.

Langley[31] procura mostrar que com o uso de circuitos tipo LSI (Large Scale Integration) a balança agora pendeu para o lado microprogramação. Ele mostra um minicomputador que utiliza LSI com a microprogramação como uma solução econômica Isso se explica pelo fato de que se pode utilizar um único fluxo de dados em inúmeras funções diferentes: controle da memória de controle, fluxo de dados propriamente dito, entrada/saída etc. Assim, o custo extra de circuitos do tipo fluxo de dados diminui. E mais, a memória de controle monolítica reuniu as duas vantagens: baixo custo e alta velocidade.

Porém quando se pensa em uma pastilha de integrado contendo toda a unidade central, a microprogramação deixa de ser uma solução viável já que a memória de controle, com circuitos auxiliares ocupariam um volume proibitivo. O que se tem feito ( INTEL ) é usar controle fixo.

Sempre que se tem duas alternativas com vantagens e desvantagens próprias, costuma-se questionar sobre uma solução intermediária; ou seja, uma unidade de controle que reuna as vantagens das duas técnicas. O modelo 145 do sistema IBM /370, o HP 2100 e outros, adotaram uma solução interessante: já que o ciclo I é sempre o mesmo,eles o fizeram fixo, deixando os ciclos de execução para serem microprograma-

dos. Ve-se então que a sequencialização do programa é feita com a técnica controle fixo e a execução por microprogramação. Aqui, pode-se ir mais longe, pode-se pensar num computador com um conjunto fixo de instruções, que seriam controladas por uma unidade de controle fixo; seriam reservados alguns códigos de instruções especiais, para serem microprogramados. Assim, poder-se-ia ter grande velocidade de execução das instruções comuns, e uma memória de controle lenta para as instruções especiais.

Pode-se encontrar além da referência número 21, um estudo bibliográfico, organizado historicamente pelo próprio autor da idéia de microprogramação: Wilkes[33].

# 4-MINICOMPUTADOR DO LSD

## 4.1 INTRODUÇÃO

De um curso de pós graduação dado pelo professor GLEN GEORGE LANGDON, JR nasceu o anteprojeto de arquitetura de um minicomputador. Os engenheiros e estagiários do Laboratório de Sistemas Digitais (LSD) terminaram o projeto e o montaram.

É um minicomputador construído com circuitos integrados da família TTL ("transistor transistor logic") e com uma memória de núcleos de ferrite.

Características Gerais

Da figura 4.1 consta um esquema do fluxo de dados e da figura 4.2 uma relação das instruções da máquina.

a) Memória Principal:
- 4 k palavras de 8 bits (núcleos de ferrite)

b) Unidade Central:
- 55 instruções
- 11 instruções com acesso à memória
- ciclo: 2 microsegundos
- endereçamento: imediato, direto e indexado
- aritmética em inteiro complemento de dois
- registrador de índice na posição zero da memória
- extensão do acumulador na posição um da memória

c) Entrada e Saída:
- por interrupção (1 nível)
- máximo de 15 dispositivos de entrada e saída

d) Outros Detalhes:
- proteção contra falhas de alimentação
- proteção das últimas 128 posiçoes de memória

e) Soffware:
- "carregador" (" Loader " ou "Bootstrap") de 128 palavras (armazenado com o pré carregador)
- programa montador (assembler)

## 4.2 DESCRIÇÃO DA ARQUITETURA

Utilizando uma memória com palavra de 8 bits, ficou definida a largura das "vias" ("busses") e circuitos tratadores de dados (somador portas de seleção, registradores, etc). Escolheu-se a representação dos números como o código complemento de dois(34) dadas suas vantagens na realização das operações aritméticas. Fez-se as "vias" de endereçamento de 12 bits necessários para o endereçamento direto das 4 K palavras da memória.

As instruções com referência à memória, são do tipo endereço simples(10), com o segundo operando operando no acumulador (registrador de 8 bits). Essas instruções são longas, ou seja, utilizam duas palavras (16 bits) já que necessita-se de 12 bits para endereçar-se o primeiro operando. A maioria das instruções sem referência à memória são de largura de 1 palavra ( 8 bits) pois não se precisa de campo de endereço.

Durante a fase de projeto lógico utilizou-se um conjunto de siglas mnemônicas para especificar as instruções. Terminado o projeto, tendo em vista á preparação dos programas em assembler, mudou-se as siglas mnemônicas para outros códigos melhores. Por isso ver-se-a, na des

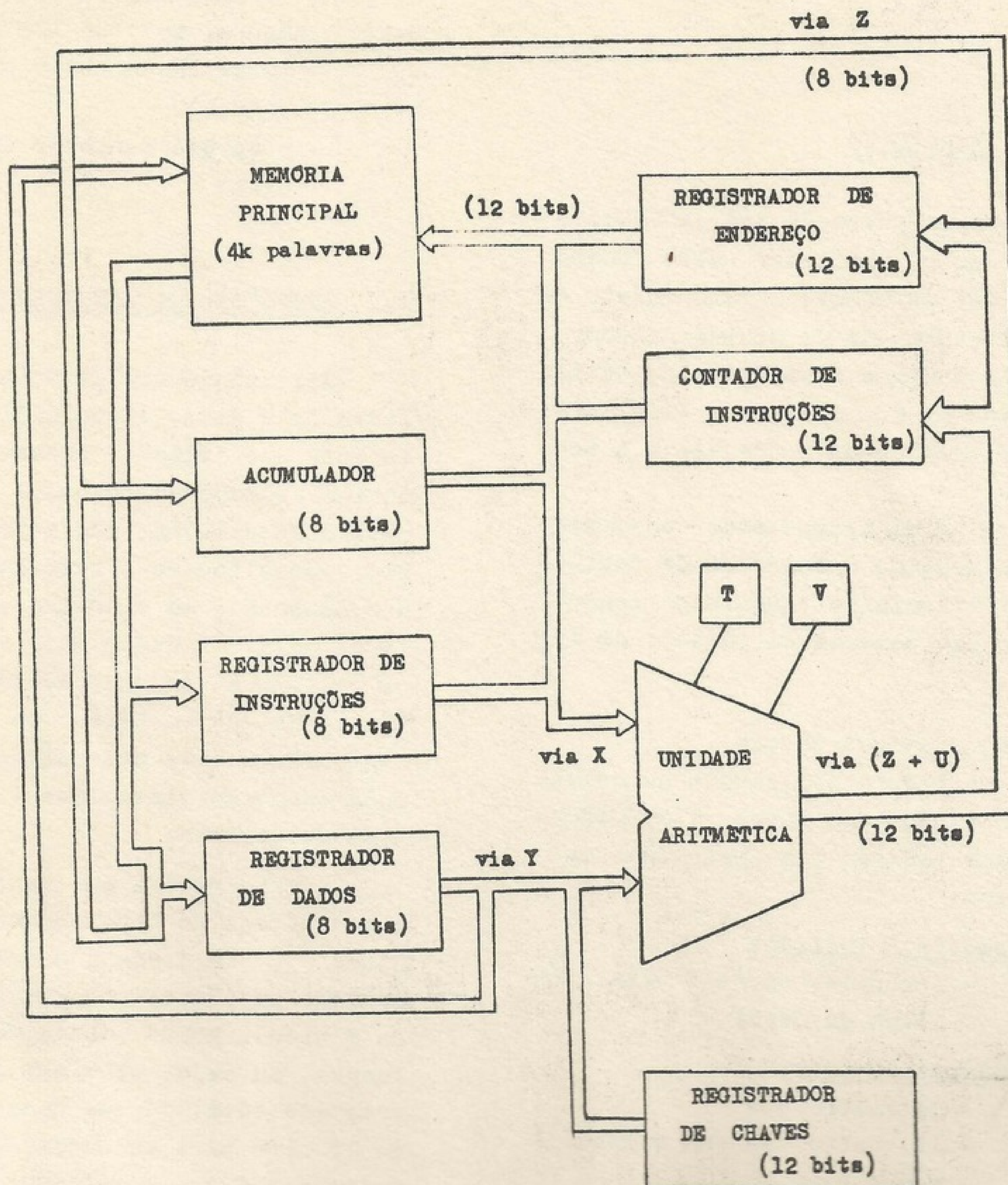



FIGURA 4.1 : Fluxo de dados simplificado

INSTRUÇÕES PRINCIPAIS

| mnemônico | descrição |
|---|---|
| SOM | soma |
| SOMX | soma indexado |
| SOMI | soma imediato |
| ARM | armazena |
| ARMX | armazena indexado |
| CAR | carrega |
| CARX | carrega indexado |
| CARI | carrega imediato |
| PUG | pula e guarda |
| SUS | subtrai um ou salta |
| NAND | nand |
| XOR | exclusive-or |
| PLA | pulo incondicional |
| PLAX | pulo indexado |
| PLAN | pula se negativo |
| PLAZ | pula se zero |
| TRE | troca com extensão |
| TRI | troca com índice |
| PUL | pula e limpa |
| PARE | pare |
| ESP | espere |
| INIB | inibe interrupção |
| PERM | permite interrupção |
| DDS | desloca à direita c/dupl.sinal |
| DD | desloca à direita |
| DE | desloca à esquerda |
| DDT | desloca à direita com T |
| DET | desloca à esquerda com T |
| GD | giro à direita |
| GE | giro à esquerda |
| GDT | giro à direita com T |
| GET | giro à esquerda com T |
| SLT | salta se T igual operando |
| SLO | Salta se OVF igual operando |
| SLTM | salta se T=operando e muda T |
| SLOM | salta se OVF=operando/muda OVF |
| LIMP | limpa AC e faz T = operando |
| CMP1 | comp. de 1 o AC e limpa T |
| CMP2 | complementa de 2 o AC |
| INC | incrementa AC |
| UM | faz AC = 1 e limpa T |
| UNEG | faz AC = -1 e limpa T |
| LIMT | limpa T |
| PNL(0) | (AC) ← (RC(4-11)), (T) ← 0 |
| PNL(1) | (AC) ← (RC(4-11)) + 1 |
| PNL(2) | (AC) ← (RC(4-11)) - (AC) - 1 |
| PNL(3) | (AC) ← (RC(4-11)) - (AC) |
| PNL(4) | (AC) ← (RC(4-11)) + (AC) |
| PNL(5) | (AC) ← (RC(4-11)) + (AC) + 1 |
| PNL(6) | (AC) ← (RC(4-11)) - 1 |
| PNL(7) | (AC) ← (RC(4-11)), (T) ← 1 |
| SAI,n,c | saída de dados p/dispositivo n |
| ENT,n,c | entr. de dados p/dispositivo n |
| SAL,n,c | salto tipo I p/dispositivo n |
| FNC,n,c | função tipo I p/dispositivo n |

Fig. 4.2 - Relação das instruções do minicomputador

crição a seguir, para cada instrução, duas siglas mnemônicas: a da primeira coluna (inicial) que aparecerá nos gráficos dos circuitos é a sigla usada na fase inicial do projeto; e a segunda coluna (definitivo) é a sigla definida para utilização pelo programador da máquina, na sua versão final.

4.2.1 INSTRUÇÕES LONGAS

São aquelas que têm o comprimento de duas palavras.

a) Grupo das instruções com referência à memória

Têm o formato visto na figura 4.3. São as instruções de endereçamento simples ( apenas um campo de endereço ).

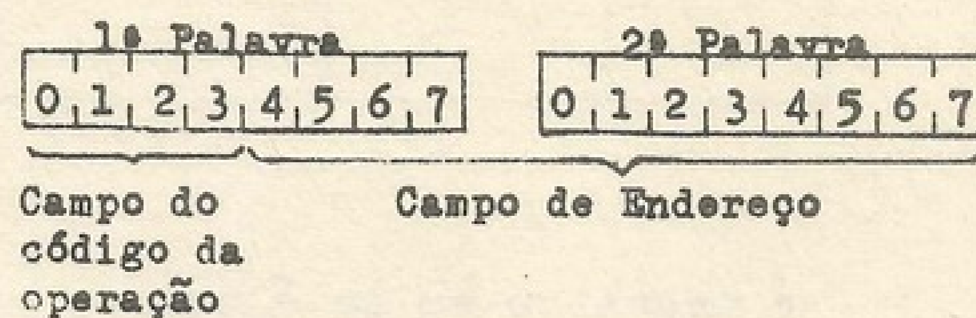



Fig. 4.3 - Formato das instruções longas com referência à memória.

Na tabela 4.1 encontra-se um quadro das instruções desse grupo.

Antes da descrição dessas instruções, é importante ressaltar a diferença entre os significados adotados para duas palavras que seriam sinônimas: pular e saltar. Pular: para nós, pular significa desviar o processamento para qualquer ponto do programa ("Jump, Branch"). Saltar: significa desviar o processamento para a instrução seguinte à próxima- saltar sôbre a próxima ("Skip").

As únicas instruções que admitem endereçamento indexado são as quatro primeiras: Pulo incondicional, Armazena ("store") Acumulador, Carrega ("Load") Acumulador e Soma no acumulador.

Existem apenas dois pulos condicionais: testando se o acumulador

TABELA 4.1

INSTRUÇÕES LONGAS COM REFERÊNCIA À MEMÓRIA

| CÓDIGO | MNEMÔNICO | | DESCRIÇÃO | OPERAÇÃO |
|---|---|---|---|---|
| | Inicial | Definitivo | | |
| 0000 | PLI | PLA | Pulo incondicional | (CI) ← END |
| 0001 | | PLAX | Pulo indexado | (CI) ← $END_{ef}$ |
| 0010 | ARA | ARM | Armazena | (END) ← (AC) |
| 0011 | | ARMX | Armazena indexado | ($END_{ef}$) ← (AC) |
| 0100 | CAC | CAR | Carrega | (AC) ← (END) |
| 0101 | | CARX | Carrega indexado | (AC) ← ($END_{ef}$) |
| 0110 | SOM | SOM | Soma | (AC) ← (AC) + (END) |
| 0111 | | SOMX | Soma indexada | (AC) ← (AC) + ($END_{ef}$) |
| 1010 | PAN | PLAN | Pula se negativo | (CI) ← (END) se AC < 0 |
| 1011 | PAZ | PLAZ | Pula se zero | (CI) ← (END) se AC = 0 |
| 1110 | SUS | SUS | Subtrai um ou salta | se (END)=0: (CI) ← (CI) + 2<br>se (END)≠0: (END) ← (END)-1 |
| 1111 | PUG | PUG | Pula e guarda | (END) ← 0000 (CI(0-3))<br>(END+1) ← (CI(4-11))<br>(CI) ← (END) + 2 |

é negativo ou se é nulo.

Fez-se a instrução subtrai- um ou salta a fim de facilitar os laços ("loops") de programa e também para testar e decrementar o registrador de índice (que é uma posição da memória).

A instrução Pula-e-Guarda é necessária para o desvio para subrotinas. Com essa instrução é feito o pulo e é montado o endereço de volta no início da subrotina.

b) Grupo de entrada e saída

São instruções utilizadas para coordenar a entrada ou saída de dados para um dos 15 dispositivos de entrada e saída. O formato dessas instruções é visto na figura 4.4.

A justificativa dessa solução para o formato das instruções é a seguinte: a primeira palavra define que é uma instrução de entrada/saída e endereça um dos 15 dispositivos possíveis com o campo n; a segunda palavra serve para definir que tipo de operação de entrada e saída se está especificando (saída, entrada, salto ou função) e dentro de cada tipo existe possibilidade de 16 diferentes, especificadas no campo comendo (c). A tabela 4.2 mostra os mnemônicos e as descrições desse grupo.

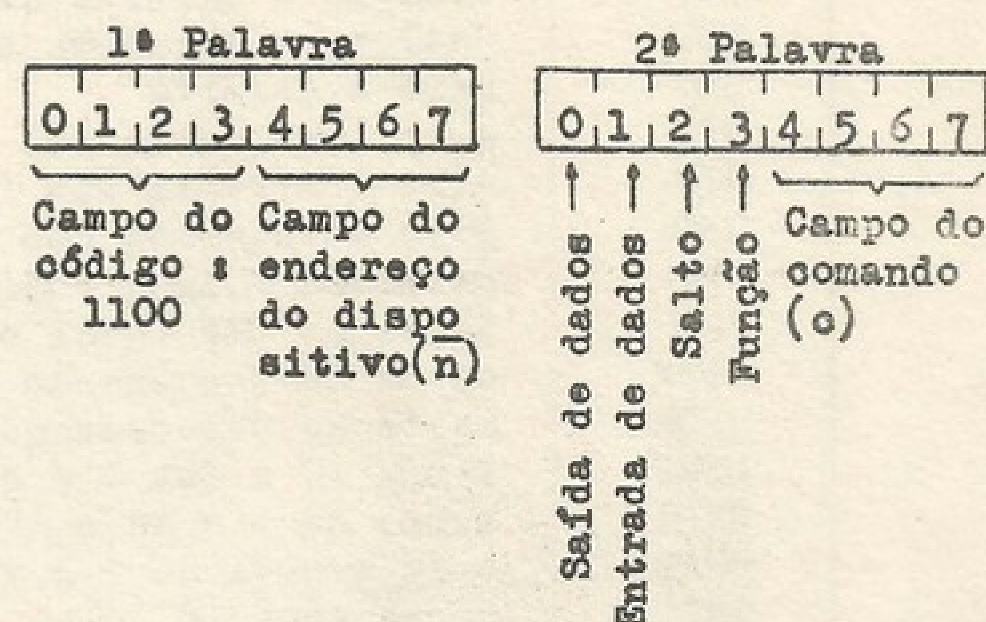



Fig. 4.4 - Formato das instruções longas do grupo entrada/saída.

TABELA 4.2

INSTRUÇÕES DO GRUPO ENTRADA E SAIDA

| MNEMÔNICO | | DESCRIÇÃO |
|---|---|---|
| Inicial | Definitivo | |
| SAEDS | SAI,n,c | saída de dados, comando c para dispositivo n |
| SAEDE | ENT,n,c | entrada de dados, comando c do dispositivo n |
| SAESA | SAL,n,c | salto, tipo c para o dispositivo n |
| SAEFU | FNC,n,c | função tipo c para o dispositivo n |

As duas primeiras instruções são óbvias. As duas últimas exigem explicações.

SAL,n,c: para cada dispositivo, dos 15 possíveis, pode-se testar alguma condição ( por exemplo, se o dispositivo está ocupado) previamente estabelecida por "Hardware" e codificada no campo Comando (portanto até 16 condições por dispositivo) e no caso dessa condição testada for satisfeita salta-se a instrução seguinte.

FNC,n,c: para cada dispositivo pode se iniciar uma dada função específica de cada um deles (por exemplo, reenrolar fita magnética), previamente estabelecida por "Hardware" e codificado no campo Comando.

c) Instruções Imediatas:

São aquelas em cujo campo de endereço existe o próprio operando. Com um campo de código das imediatas de 4 bits, poder-se-ia implementar até 16 instruções nesse grupo. Contudo, bastaram 4 delas: Soma, Nand, Exclusive-or e Carrega acumulador. Aproveitou-se então para escolher-se os códigos de modo a simplificar o decodificador. E ainda colocou-se, junto com essas instruções, a instrução de deslocamento que não é imediata. Veja na figura 4.5 o formato das instruções imediatas e na 4.6 o das instruções de deslocamento.

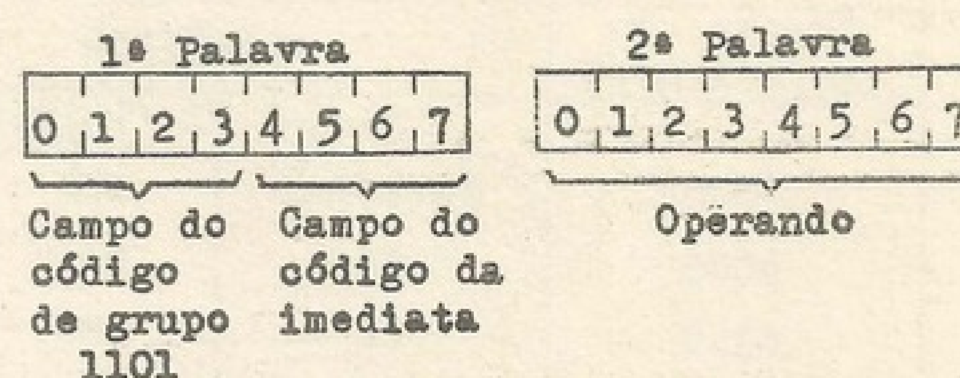



Fig. 4.5 - Formato das instruções imediatas.

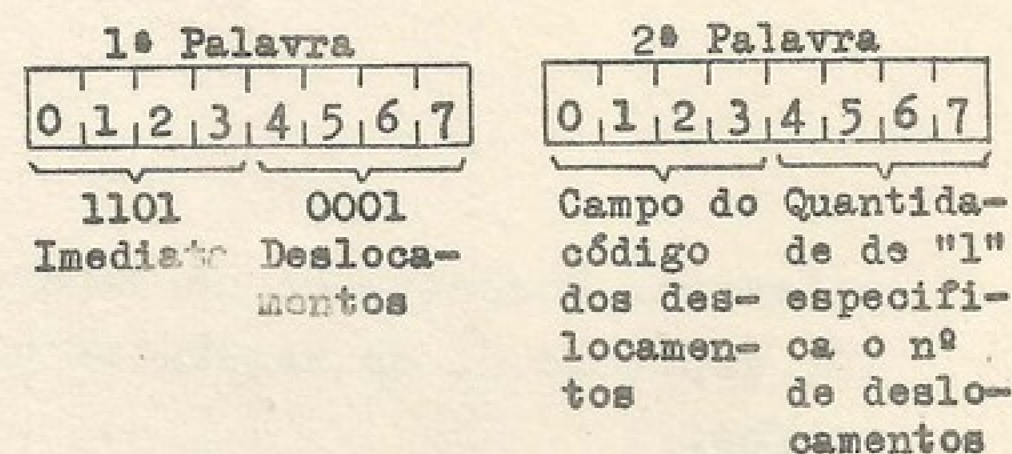



Fig. 4.6 - Formato das instruções de deslocamento.

Nas tabelas 4.3 e 4.4 encontram-se, respectivamente, detalhes sobre as instruções imediatas e as instruções de deslocamento.

Sobre as instruções imediatas dispensa-se explicações. Quanto as instruções de deslocamento, que envolve dois registradores (acumulador de 8 bits e vai-um de 1 bit), acredita-se ser necessário algumas

TABELA 4.3

INSTRUÇÕES IMEDIATAS

| CÓDIGO DA IMEDIATA | MNEMÔNICO | | DESCRIÇÃO | OPERAÇÃO |
|---|---|---|---|---|
| | Inicial | Definitivo | | |
| 1000 | IMEADD | SOMI | Soma imediata | (AC) ← (AC) + Operando |
| 0100 | IMENAND | NAND | NAND imediato | (AC) ←((AC) ∧ Operando)' |
| 0010 | IMEXOR | XOR | EXCLUSIVE-OR imediato | (AC) ← (AC) ⊕ Operando |
| 1010 | IMECAC | CARI | Carrega imediato | (AC) ← Operando |

TABELA 4.4

INSTRUÇÕES DE DESLOCAMENTO

| CÓDIGO DO DESLOCAMENTO | MNEMÔNICO | | DESCRIÇÃO |
|---|---|---|---|
| | Inicial | Definitivo | |
| 1000 | SR+DUPSI | DDS | Desloca à direita com duplicação do sinal |
| 0000 | SR | DD | Desloca à direita |
| 0100 | SL | DE | Desloca à esquerda |
| 0001 | SR COM T | DDV | Desloca à direita com V |
| 0101 | SL COM T | DEV | Desloca à esquerda com V |
| 0010 | SR COM G | GD | Giro à direita |
| 0110 | SL COM G | GE | Giro à esquerda |
| 0011 | SR C/T e G | GDV | Giro à direita com V |
| 0111 | SL C/T e G | GEV | Giro à esquerda com V |

palavras. Encontra-se na referência 10, capítulo 5, os seguintes parágrafos:

"Os giros e deslocamentos funcionam da seguinte maneira: o flip-flop V do acumulador sempre recebe o transbordo ou da posição direita (bit 7) para deslocamento ou giros à direita, ou da posição esquerda (bit 0) para deslocamentos ou giros à esquerda".

"Nos deslocamentos simples à direita ou à esquerda (DD ou DE) o vai-um transbordo vai para o registrador V e o outro extremo do acumulador é preenchido com zeros".

"Os giros simples à direita ou à esquerda, (GD ou GE) o vai-um transbordo além de ir para o registrador V é realimentado para o outro extremo do acumulador, que neste caso não é preenchido com zeros. Aqui, o acumulador funciona como um anel, um extremo ligado ao outro".

"As instruções DDV e GDV são iguais. O mesmo se diz dos DEV e GEV. Nessas instruções o acumulador se fecha, também como um anel, tendo o registrador V intercalado entre os extremos dele. E o deslocamento se processa em giro, à esquerda ou à direita".

"Na instrução particular DDS (desloca à direita com duplicação de sinal), o bit "0" (sinal do acumulador) não muda e ocorre desloca-

mento à direita. Resulta então uma duplicação desse sinal para dentro do acumulador conforme os deslocamentos vão ocorrendo".

4.2.2 INSTRUÇÕES CURTAS

As instruções curtas são instruções de apenas uma palavra. O computador distingue as instruções curtas das longas pelos três primeiros bits, que nas curtas devem ser: 100. Qualquer instrução cujos três primeiros bits da primeira palavra não formem o número 100 é uma instrução longa. Nessa arquitetura a instrução curta recebeu o nome de Micro Instrução. É importante ressaltar que esse nome nada tem a ver com microprogramas, foi dado esse nome porque é uma instrução curta. Essas instruções curtas são divididas em dois grupos:

a) Micros grupo 1 (Mic G 1)

Na figura 4.7 encontra-se o formato desse grupo de microinstruções. É importante observar que cada bit (dos 4 últimos) especifica sinais elétricos diretamente no fluxo de dados. Outra coisa, dada a implementação física das portas de seleção, a presença simultânea dos sinais, que controlam a entrada da unidade aritmética, AC/X e $\overline{AC}$/X leva todos os bits na entrada X para o nível "1". Na tabela 4.5 vê-se uma descrição das microoperações obtidas com a combinação desses sinais.

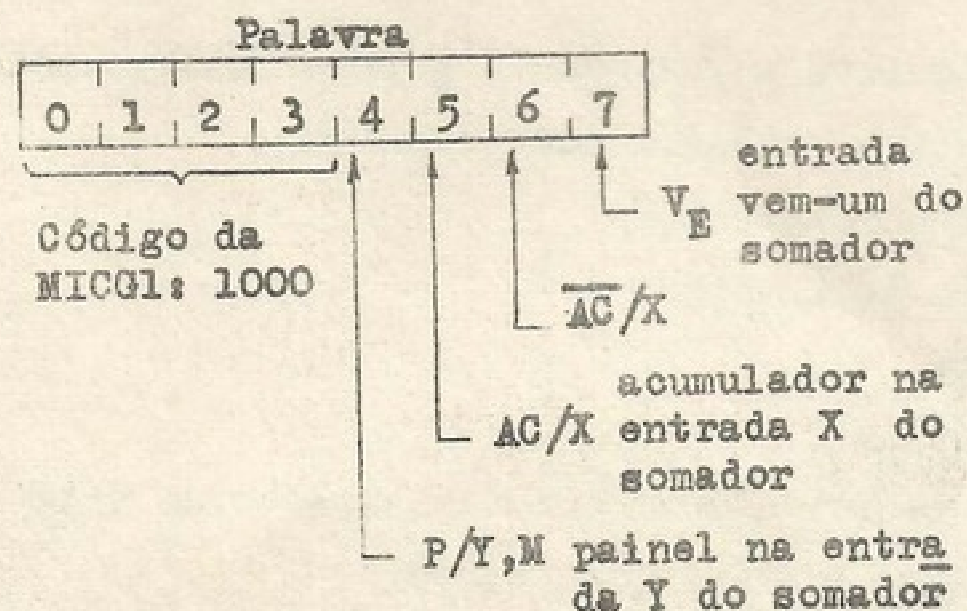



Fig. 4.7 - Formato das instruções do grupo 1

É importante frizar que nesse grupo, a saída do somador é colocado no acumulador. O que as instruções fazem é controlar as entradas do somador. Uma grande vantagem dessas microinstruções é a velocidade: a execução delas ocorre durante o próprio ciclo I, ou seja, ao todo elas gastam apenas um ciclo, 2 micro segundos.

b) Micros Grupo 2

São divididas em dois subgrupos: as instruções de saltos condicionais e as especiais.

Micros Grupo 2 A (MICG2A)

São usadas para testar os estados dos flip flops V (vai-um) e T (transbordamento) da unidade aritmética. Na figura 4.7 ve-se o seu formato e na tabela 4.6 sua descrição.

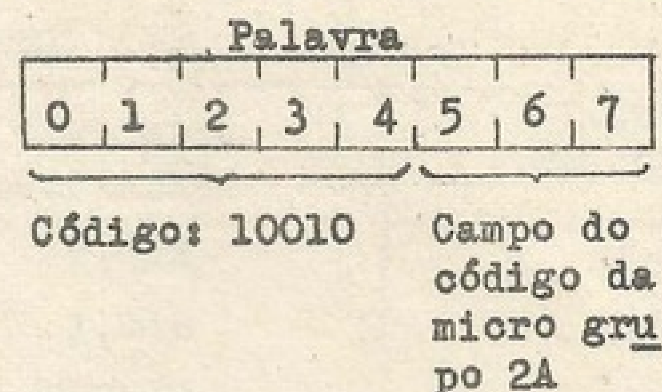



Fig. 4.8 - Formato das micro grupo 2A

Os saltos são executados (incluído ciclo I) em um ciclo da máquina apenas. Essas instruções seguidas de um pulo incondicional, equivalem a Pulos Condicionais, testando os flip flops V e T.

Micros Grupo 2 B (MICG2B)

É um grupo de instruções com o mesmo formato das MICG2A (veja fig. 4.9). A tabela 4.7 apresenta a relação dessas instruções.

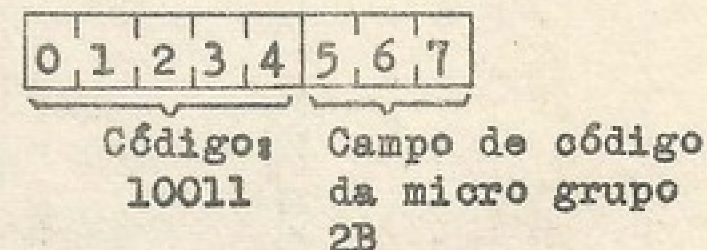



Fig. 4.9 - Formato das micro grupo 2B

Aqui, torna-se necessária uma explicação. Aproveitou-se novamente

TABELA 4.5

RELAÇÃO DAS MICROINSTRUÇÕES DO GRUPO 1

| INSTRUÇÃO | MNEMÔNICO Definitivo | DESCRIÇÃO |
|---|---|---|
| 1000 0000 | LIMP,0 | Limpa AC e faz V = 0 |
| 1000 0111 | LIMP,1 | Limpa AC e faz V = 1 |
| 1000 0001 | UM | Faz AC = 1 e limpa V |
| 1000 0010 | CMP1 | Complementa de 1 o acumulador e limpa V |
| 1000 0011 | CMP2 | Complementa de 2 o acumulador e limpa V |
| 1000 0100 | LIMV | Limpa V |
| 1000 0101 | INC | Incrementa o acumulador |
| 1000 0110 | UNEG | Faz acumulador = -1 e limpa T |
| 1000 1000 | PNL(0) | (AC) ← (RC(4-11)), (V) ← 0 |
| 1000 1001 | PNL(1) | (AC) ← (RC(4-11)) + 1 |
| 1000 1010 | PNL(2) | (AC) ← (RC(4-11)) - (AC) - 1 |
| 1000 1011 | PNL(3) | (AC) ← (RC(4-11)) - (AC) |
| 1000 1100 | PNL(4) | (AC) ← (RC(4-11)) + (AC) |
| 1000 1101 | PNL(5) | (AC) ← (RC(4-11)) + (AC) + 1 |
| 1000 1110 | PNL(6) | (AC) ← (RC(4-11)) - 1 |
| 1000 1111 | PNL(7) | (AC) ← (RC(4-11)), (V) ← 1 |

TABELA 4.6

INSTRUÇÕES DO GRUPO MICG2A

| CÓDIGO DA MICG2A | MNEMÔNICO Definitivo | DESCRIÇÃO |
|---|---|---|
| 000 | SLV,0 | Salta se V igual a 0 |
| 001 | SLV,1 | Salta se V igual a 1 |
| 010 | SLVM,0 | Se V igual a 0, salta e faz V igual a 1 |
| 011 | SLVM,1 | Se V igual a 1, salta e faz V igual a 0 |
| 100 | SLT,0 | Salta se T igual a 0 |
| 101 | SLT,1 | Salta se T igual a 1 |
| 110 | SLTM,0 | Se T igual a 0, salta e faz T igual a 1 |
| 111 | SLTM,1 | Se T igual a 1, salta e faz T igual a 0 |

TABELA 4.7

INSTRUÇÕES DO GRUPO MICG2B

| CÓDIGO DA MICG2B | MNEMÔNICO | | DESCRIÇÃO |
|---|---|---|---|
| | Inicial | Definitivo | |
| 000 | PUL | PUL | Pula para a posiçao 2 e limpa interrupção |
| 001 | TRE | TRE | Troca acumulador com extensão |
| 010 | INI | INIB | Inibe interrupção |
| 011 | PER | PERM | Permite interrupção |
| 100 | PAR-D | PARE | Pare |
| 101 | ESP-D | ESP | Espere |
| 110 | TRO | TRI | Troca acumulador com registrador de índice |
| 111 | **** | **** | Reservado para uso futuro |

alguns parágrafos da referência 10:

"Ressaltamos que as posições 0 e 1 da memória são, respectivamente o indexador e a extensão do acumulador. Instruções "TRI" e"TRE" permitem os meios econômicos para retirar ou carregar estas palavras. Essas instruções trocam o conteúdo da respectiva palavra com o conteúdo do acumulador. As outras microinstruções do grupo 2A são vinculadas com o esquema de interrupção, um assunto abordado mais adiante. Brevemente, "INIB" inibe o sistema de interrupção, não permitindo o programa em andamento ser interrompido, até depois da execução da instrução "PERM". Quando um programa é interrompido, o "CI" fica armazenado nas posições 2 e 3 da memória, com os 4 bits mais significativos da palavra 2 sendo "0000". Assim, a execução de "PUL" devolve o controle ao programa que foi interrompido."

"A instrução "ESP" coloca o computador num estado de "espera" até que seja acionado o botão de partida do painel ou uma interrupção acontece. A instrução "PARE" é igual a "ESP" com a exceção que "PARE" não aceita interrupção, o computador só continua com intervenção pelo painel."

## 4.3 DESCRIÇÃO DO FLUXO DE DADOS

A figura 4.1 mostrou um esquema simplificado do fluxo de dados do minicomputador. A figura 4.10 apresenta outro esquema um pouco mais detalhado.

### 4.3.1 ANÁLISE EM BLOCOS

Na figura 4.10 distingue-se os seguintes blocos funcionais: memória, registradores, unidade aritmética e portas de seleção. Na referência 10 encontra-se informações gerais sôbre esses blocos.

a) Memória

A memória principal, organizada em palavras de 8 bits num total de 4096 palavras, é uma memória de núcleos de ferrite, numa montagem de 3 fios por núcleo (FI 21 - Philips[35]). Dos sete modos de funcionamento, ressaltar-se-á os três usados:

ciclo completo:

Ler e restaurar (M4):
- duração total: 1,6 μseg.
- tempo de acesso: 0,4 μseg.

ciclo rachado:

Ler (M1):
- duração total: 0,6 μseg.
- tempo de acesso: 0,4 μseg.

Escrever (M3):
- duração total: 1,0 μseg.

Sob o ponto de vista da unidade de controle, é preciso lembrar que, durante todo o ciclo, o endereço deve ficar constante. Existem inúmeros sinais elétricos necessários para controlar a memória, e por isso projetou-se um circuito de interface de tal forma que quando a unidade de controle precisar ler ou escrever algo na memória ela envia apenas os sinais M4 ou M1 ou M3 que esse circuito gera os controles necessários.

As posições 0 e 1 da memória são reservadas como índice e extensão do acumulador, respectivamente.

b) Registradores

Os cincos registradores do fluxo de dados podem ser agrupados em duas classes: os registradores de dados (RD, AC, RI) são de 8 bits com a finalidade de armazenar uma palavra de memória; e os registradores de endereço (RE e CI) com o objetivo de guardarem endereços. Todos os registradores foram implementados com flip flops tipo D sensí-

Figura 4.10 : Fluxo de dados mais detalhado

veis à borda[10]. Segue uma breve descrição desses registradores:

Registrador de Endereço (RE):

É o registrador que tem a função de segurar o endereço de memória e mantê-lo fixo durante todo o ciclo. Sua entrada são vias Z e U - saídas da unidade aritmética, e sua saída alimenta a memória através da porta de seleção 5. Esse registrador tem uma linha de controle ("set reset") para forçar o endereço 2 - utilizado quando chega um pedido de interrupção.

Contador de Instruções (CI):

É outro registrador de 12 bits com a entrada ligada à saída da unidade aritmética. Sua função é armazenar o endereço da instrução seguinte (note-se que nesta arquitetura este registrador faz parte do fluxo de dados e não da unidade de controle). Para incrementar um ao endereço utiliza-se a unidade aritmética, fazendo-se a soma com o VUQ ("vai-um-quente" ou vem-um). A saído do CI vai direto à entrada da unidade aritmética.

Acumulador (AC):

É um registrador deslocador de 8 bits, com a opção de colocar o flip flop V (vai-um) em série com os 8 bits como se fosse um nono bit do acumulador. Esse registrador é, talvez, o núcleo da arquitetura do computador: todas as operações aritméticas e lógicas são feitas tendo o conteúdo do acumulador como um dos operandos e o resultado sendo armazenado nele: todos os desvios condicionais são feitos a partir de testes em seu conteúdo; a entrada e saída de dados são feitas através dele.

Registrador de Dados da Memória (RD)

É um registrador de 8 bits que serve com um "buffer" entre a UCP e a memória. Dados lidos na memória são armazenados nele, assim como dados para gravar na memória é nele que ficam durante o ciclo de escrita. Por esse motivo o 2º operando das instruções aritméticas e lógicas são armazenados nele (veja, sua saída está ligada na entrada Y da unidade aritmética), já que o 1º, conforme já foi dito, é o conteúdo do acumulador.

Registrador de Instruções (RI):

Como a maioria das instruções desse computador são longas, isto é de duas palavras, o RI é usado para armazenar a 1a. palavra da instrução enquanto que a 2a. palavra (endereço do operando) fica no RD até a leitura do operando. É um registrador de 8 bits provido de uma linha de "set" que permite forçar a instrução PUG (código 1111) necessária durante a interrupção por um dispositivo de entrada/saída.

c) Unidade Aritmética

A função básica da unidade aritmética é a realização das operações aritméticas e lógicas especificadas pelas instruções. São elas: soma, nand e exclusive-or com operandos de 8 bits. A figura 4.10 mostra um bloco denominado "unidade aritmética" que realiza essas funções. Esse bloco tem largura de 8 bits.

Porém, a unidade aritmética precisa executar uma outra função: operar em endereços, ou somando o índice ao endereço, ou somando 1 ao contador de instruções. Por isso, inclui-se em paralelo com a unidade aritmética um circuito chamado "mais um" ("one-upper") que trata dos 4 bits mais significativos dos endereços. Com essa providência consegue-se somar um número de 12 bits (endereço) com um de 8 bits (índice).

A estrutura física da Unidade Aritmética é do tipo somador com propagação do vai- um(10) do (ripple carry"(37) " carries ripple throught"(38)), com um atraso de 450 nseg no pior caso para terminar a soma com 12 bits.

d) Portas de Seleção

Na figura 4.11 encontra-se o esquema de uma porta de seleção. Esse circuito é usado sempre que existem duas (ou mais) palavras, distintas, possíveis de serem colocadas na entrada de um circuito qualquer. Essas palavras são colocadas nas entradas da porta de seleção cuja saída é ligada na entrada do circuito citado. Veja na figura 4.11.




**Fig. 4.11 - Esquema de uma porta de seleção.**

Esse circuito tem o seguinte comportamento: com E1/S e E2/S no nível "zero", a saída S é nula. Quando E1/S fica igual a "um" a via de entrada E1 é colocada na via de saída S. Com E2 é análogo. Detalhes sobre portas de seleção são encontrados na referência 10. Nesta arquitetura, existem 6 portas de seleção:

Porta de Seleção 1

Controla a entrada do circuito "mais-um". Permite que se coloque na via M as seguintes palavras:

RI(4-7) - para endereçar o operando
CI(0-3) - para somar um ao CI ou transportar CI para RE
RE(0-3) - para somar o índice de registro ao endereço
RC(0-3) - para endereçar pelo painel

Porta de seleção 2

Controla a entrada Y do somador. Seleciona entre duas palavras:

RD - nas operações aritméticas
RC(4-11) - para introduzir dados através das chaves do painel, ou junto com a porta de seleção 1 para endereçar pelo painel.

Porta de seleção 3

Controla a entrada X do somador. Na maioria das vezes funciona junto com a porta de seleção 1 ou 2 Seleciona uma das 4 seguintes entradas:

AC - para operar com o acumulador
$\overline{AC}$ - para complementar o acumulador
RE(4-11) - para somar o índice ao endereço
CI(4-11) - para somar um ao CI, ou transportar CI para RE.

Porta de seleção 4

Controla a entrada de RD, selecionando uma das seguintes três entradas:

VIA Z - para operandos gravados na memória
LWbo - para palavras lidas na memória
VIA U - utilizada pela instrução de PUG.

Porta de seleção 5

Controla o endereçamento da memória. Possibilita as seguintes três situações:

saída zero: para endereçar o índice de registro
saída um: para endereçar a extensão do acumulador
saída RE: nos endereçamentos normal, onde a memória é endereçada pelo RE.

Porta de seleção 6

Controla a entrada do acumulador, selecionando entre via Z (fun-

cionamento normal) ou via E (caso de entrada de dados).

### 4.3.2 ANÁLISE DOS CIRCUITOS

Particionou-se os circuitos do fluxo de dados em 8 placas de circuito impresso, além da memória. Na partição, levou-se em conta inúmeros vínculos como quantidade máxima de pastilhas de circuitos impressos por placa, número máximo de entradas e saídas da placa, etc. O critério de partição foi o da economia de pinos nos conectores, ou seja tenta-se atingir o máximo de circuitos com um mínimo de entrada e saída da placa. Outro fator importante foi tentar-se, à medida do possível fazer placas iguais para se economizar na preparação do "layout" e arte final do circuito impresso. A partição resultou nas seguintes placas:

| Nº da placa | Descrição |
|---|---|
| FR1-1 : | registradores- bits pares |
| FR2-2 : | registradores- bits impares |
| FS1-3 : | somador- bits menos significativos |
| FS2-4 : | somador- bits mais significativos |
| FAC-5 : | acumulador |
| FM1-6 : | mais-um e porta de seleção 6 |
| FCM-7 : | Interface com memória |
| FIN-8 : | sinais para cartões de interface. |

São esses os 8 cartões que compõem o fluxo de dados. Segue uma breve descrição de cada um deles. Nos esquemas apresentados a seguir, linhas cheias são de dados e linhas pontilhadas de controle.

a) FR1-1 e FR2-2

São duas placas de circuito impresso iguais: a primeira com os bits das posições pares e a segunda das posições ímpares. Na figura 4.12 encontra-se um esquema do circuito dessa placa. Observe-se que ela inclui os registradores: CI, RE com sua linha de controle ("set-reset") para forçar 2, RI com outra linha de controle ("set") para forçar o PUG e RD. Há ainda, nessa placa, as portas de seleção 4 e 5. Na prancha I encontra-se o circuito detalhado.

b) FS1-3 e FS2-4

São dois cartões iguais contendo os circuitos da Unidade Aritmética. O primeiro corresponde aos bits menos significativos e o segundo aos mais significativos. Na figura 4.13 encontra-se o esquema em bloco de seu conteúdo: portas de seleção 2 e 3 e unidade aritmética. Na prancha II há os detalhes desse circuito.

c) FAC-5

A FAC-5 é um registrador deslocador: o acumulador. Colocou-se o acumulador numa só placa, junto com os flip flops V e T, já que sua estrutura não é uniforme a ponto de ser "quebrada em duas" iguais. A figura 4.14 mostra a placa em blocos e a prancha III em detalhes.

d) FM1-6

É a placa que tem os circuitos da porta de seleção 6, porta de seleção 1, circuito mais-um, e geração do transbordamento para o flip-flop T ("overflow"). Na figura 4.15 vê-se um esquema e na prancha IV os circuitos detalhados. Esta placa contém uma miscelânea já que nela foram colocados os pedaços do fluxo de dados que não couberam em outro lugar.

e) FCM-7

É a placa que gera os controles da memória. Suas funções são as seguintes: endereçar corretamente os módulos da memória (são 4); coor

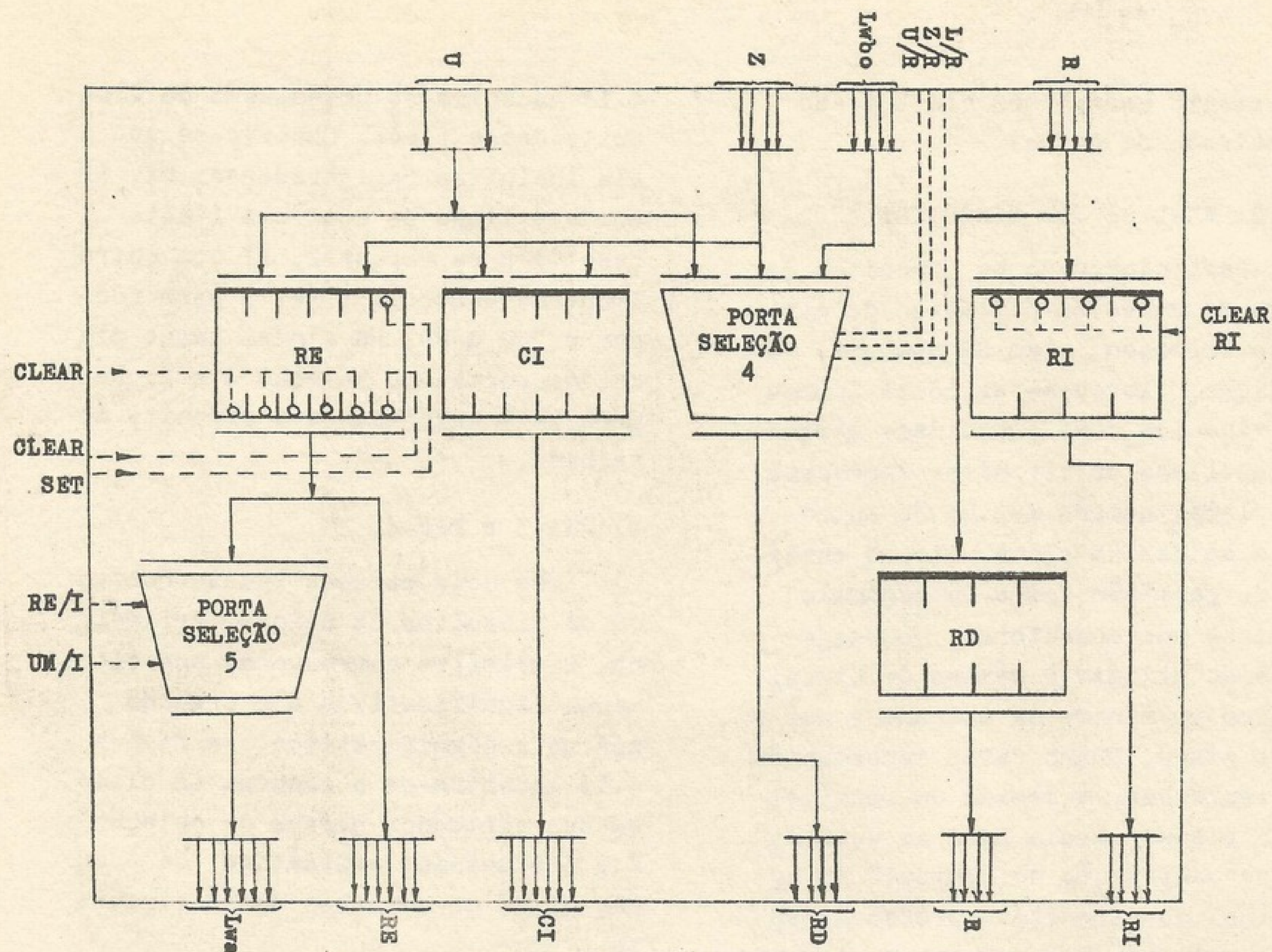



Fig. 4.12 - Cartão dos registradores (FR1-1, FR2-2)

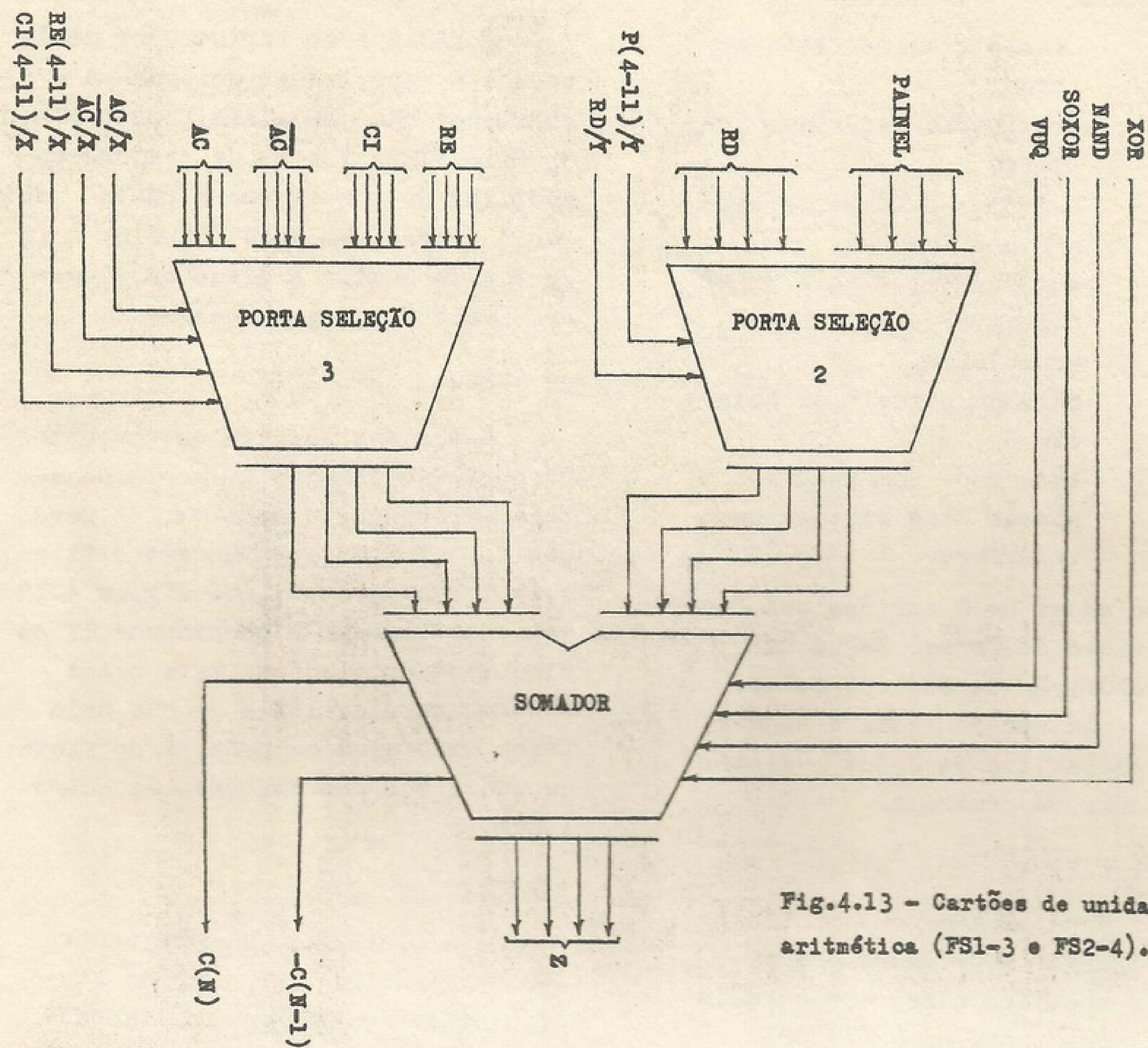



Fig.4.13 - Cartões de unidade aritmética (FS1-3 e FS2-4).

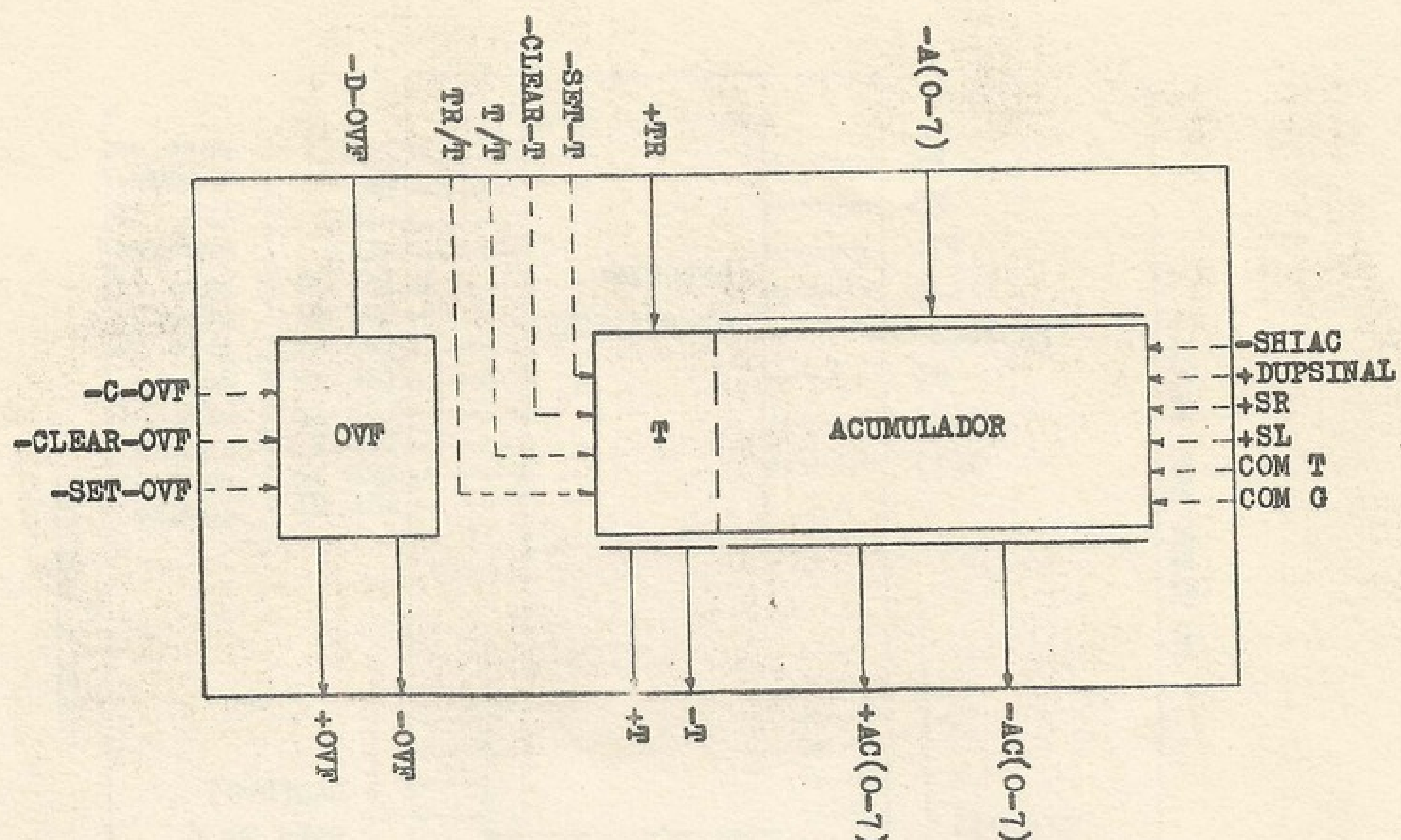



Fig. 4.14 - Esquema da placa do acumulador (FAC-5)

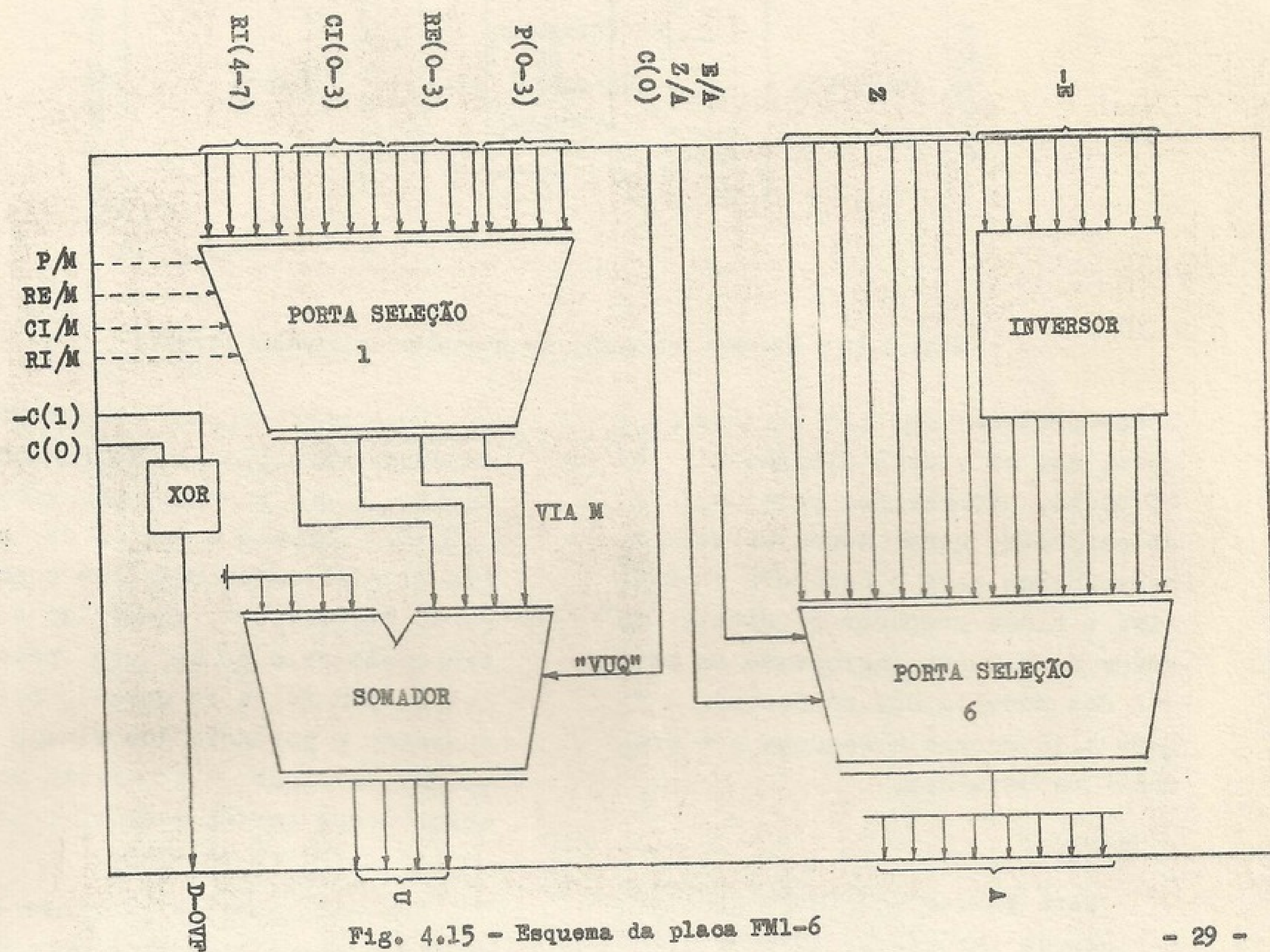



Fig. 4.15 - Esquema da placa FM1-6

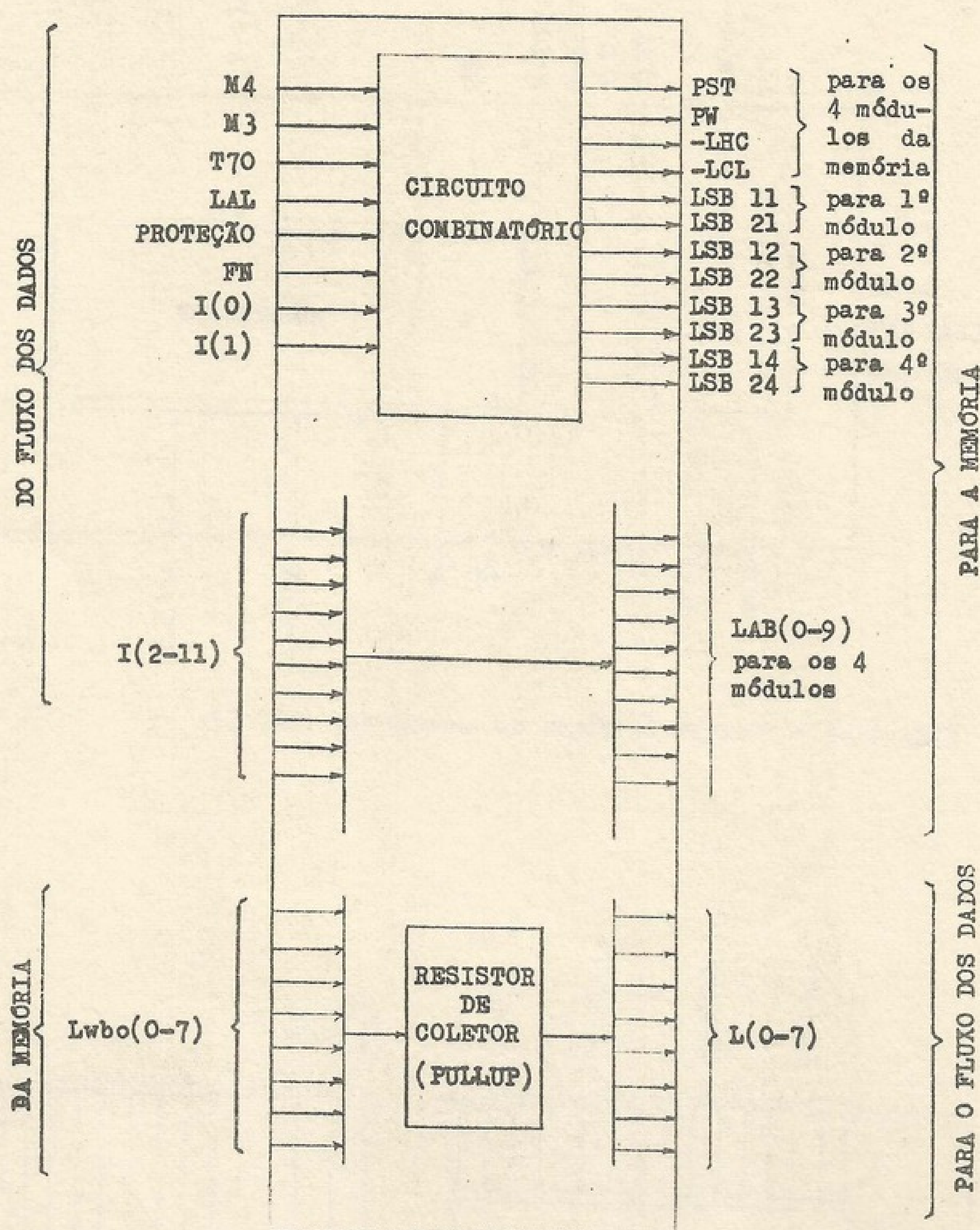



Fig. 4.16 - Esquema do cartão de contrôle de memória (FCM-7)

denar proteção das 128 últimas posições; com os sinais simples M1 ou M3 ou M4, fornecidos pela unidade de controle, gerar todos os sinais necessários para o controle da memória; e ainda preparar os sinais de saída da memória (agrupando as saídas dos módulos num só ponto). A figura 4.16 mostra o esquema e a prancha V os detalhes.

f) FIN-8

Para enviar os sinais para os cartões de interface, deve-se fazer com que as vias ("busses") passem por portas que se abrem apenas quando é uma instrução de entrada e saída. Isso para evitar que existam sinais nas vias quando não for necessário, já que esses sinais gerariam ruídos. Essa é a funçaõ da placa FIN-8, além de decodificar o endereço do dispositivo, controlar a interrupção pelo canal zero (pelo painel ou por falta de força), além de aumentar a potência dos sinais de saída. Na figura 4.17 existe um esquema desse cartão e na prancha VI os circuitos em detalhes.

Os oito cartões são interligados pelo painel trazeiro do computador formando um só bloco: o fluxo de dados.

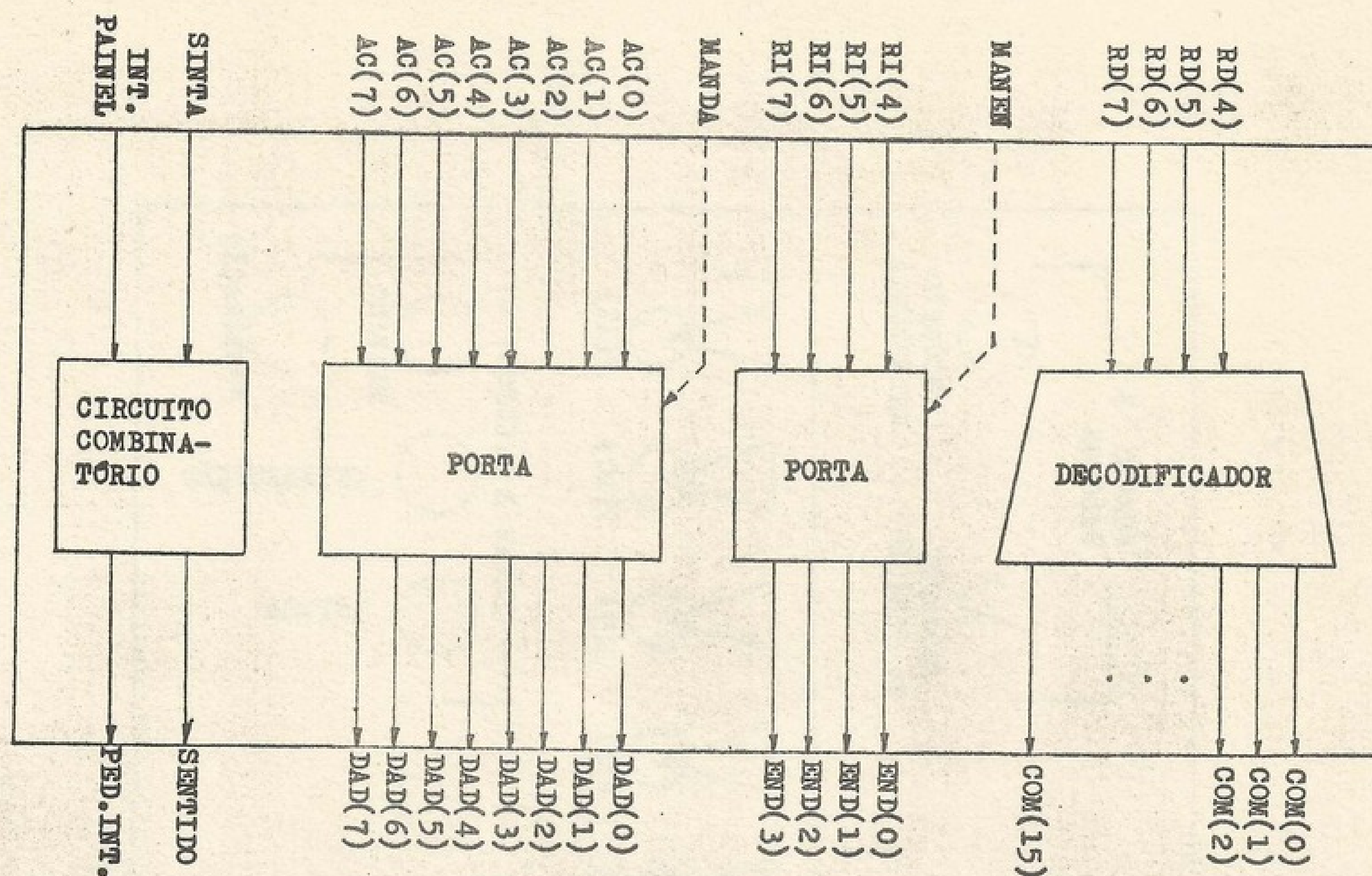



Fig. 4.17 - Esquema do cartão dos sinais de interface (FIN-8)

## 4.4 OUTROS DETALHES

### 4.4.1 INTERRUPÇÃO

"É frequentemente desejável dar ao computador a habilidade de responder a estímulos outros que aqueles gerados pelas operações correntes, internas à máquina"(11). A interrupção dá essa capacidade ao sistema.

A rotina de atendimento aos pedidos de interrupção começa na posição 2 de memória. Quando chega um pedido de interrupção, é feito, por "hardware", um PUG para a posição 2. Então nas posições 2 e 3 da memória ficará armazenado o endereço de volta ao programa principal e o processamento iniciará na instrução armazenada em 3 e 4 que é um pulo para a rotina tratadora da interrupção.

No final da rotina tratadora da interrupção existe uma instrução de PUL, que corresponde a um pulo para a posição 2 com liberação do estado de interrompido. Na posição 2 encontrará um pulo de volta para o ponto de onde saiu do programa principal.

O tratamento da instrução de PUL, quanto à liberação do estado interrompido, é peculiar. Se essa liberação ocorrer durante o ciclo E da própria PUL, existe a possibilidade de ocorrer uma nova interrupção nesse ponto, que fará com que seja montado o endereço da instrução seguinte à PUL em cima do endereço de volta ao programa principal. E então se perderá o ponto de volta. Para se evitar isso, a unidade de controle deve limpar o estado interrompido no ciclo seguinte ao ci-

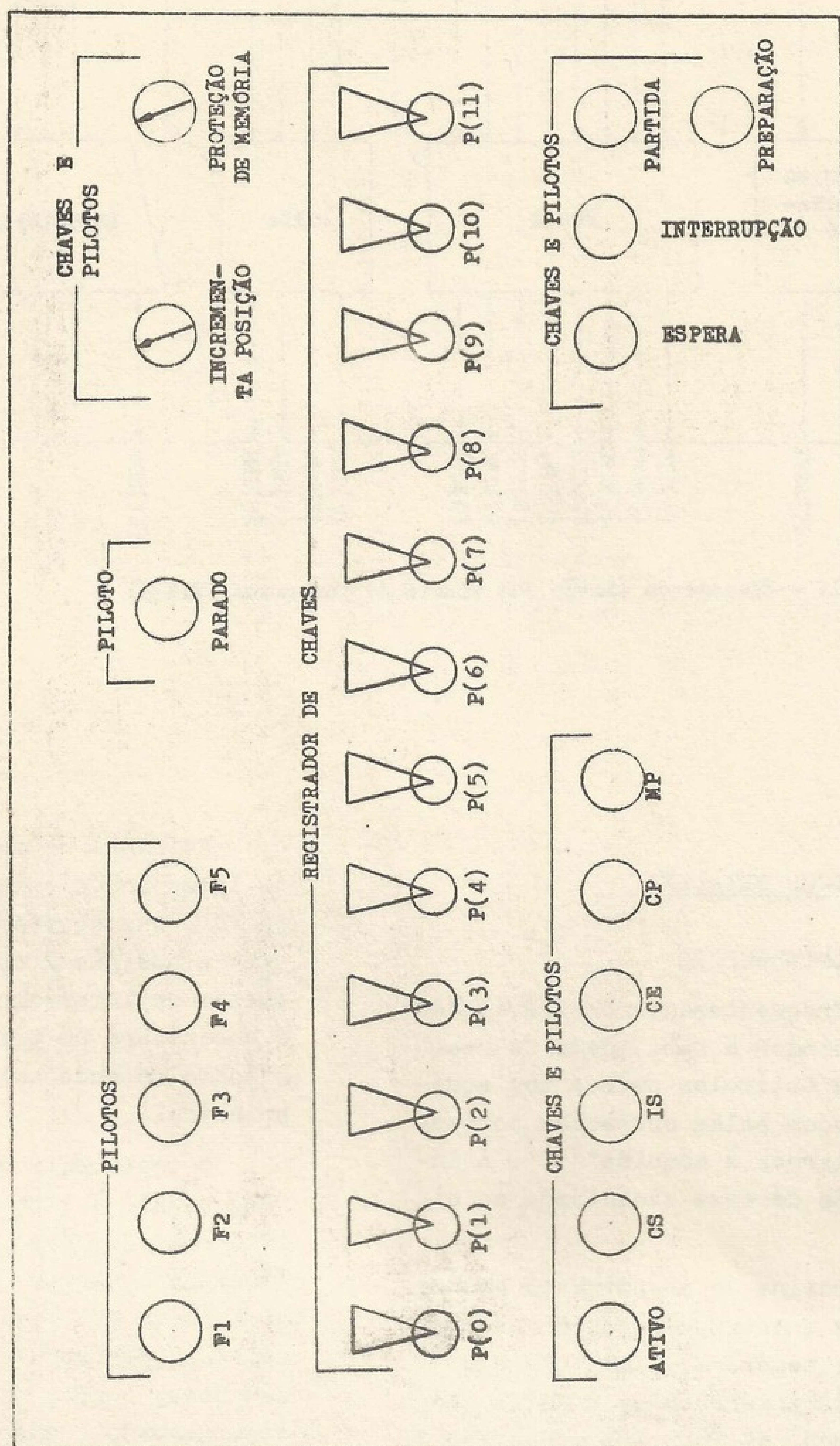



FIGURA 4.18 : CONTROLES DO PAINEL

clo E da instrução de PUG. Detalhes gerais sobre interrupção encontram-se nos itens 10 e 11 da referência.

4.4.2 CONTROLES PELO PAINEL

No painel existem chaves que permitem ao operador agir sobre o sistema Existem lâmpadas pilotos que informam ao operador o estado do sistema. Existem LED's ("light emitting diodes") que indicam o conteúdo dos registradores centrais (veja figura 4.18). Duas placas de circuito impresso realizam o trabalho de comunicação entre a UCP e o painel:

Placa PMR-1 - vista na figura 4.19 em blocos e na prancha VII em detalhes. Essa placa tem os drivers dos LED's.

Placa PCP-2 - tem os circuitos eliminadores de ruido ("debounce") das chaves e os drivers para as lâmpadas piloto.

Pode-se classificar as chaves do painel em dois tipos: a) controle de ação e b) controle de modo de operação

a) Controle de ação

São as chaves que mudam o estado do sistema. São elas:

PARTIDA: quando o sistema está no estado parado ou esperando, ela inicia o seu funcionamento.

ESPERE: faz com que a máquina pare o seu funcionamento normal e entre no estado de espera.

INTERROMPE: envia um pedido de interrupção pelo canal "zero". A ação que ocorrer a seguir depende do programa tratador de interrupção pelo canal zero.

b) Controle do modo de operação

Existem seis chaves para selecionar os seis modos de funcionamento, mutuamente exclusivos: normal, ciclo único, instrução única, carrega endereço, carrega posição e mostra posição. Durante a fase de projeto deu-se a esses modos os seguintes nomes: ativo, ciclo simples (CS), instrução simples (IS),

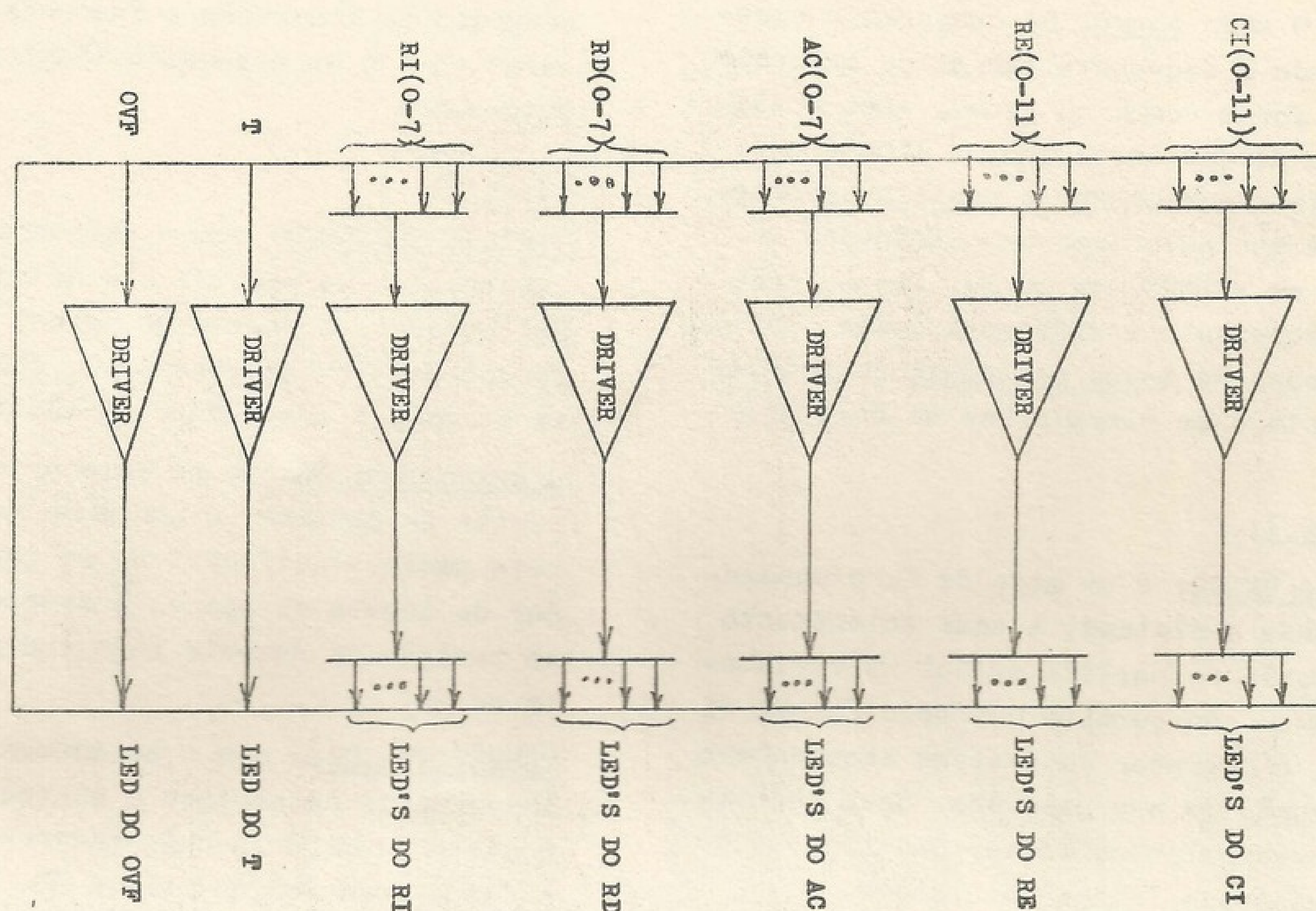



Fig. 4.19 - Esquema do cartão PMR-1 (drivers dos LED'S)

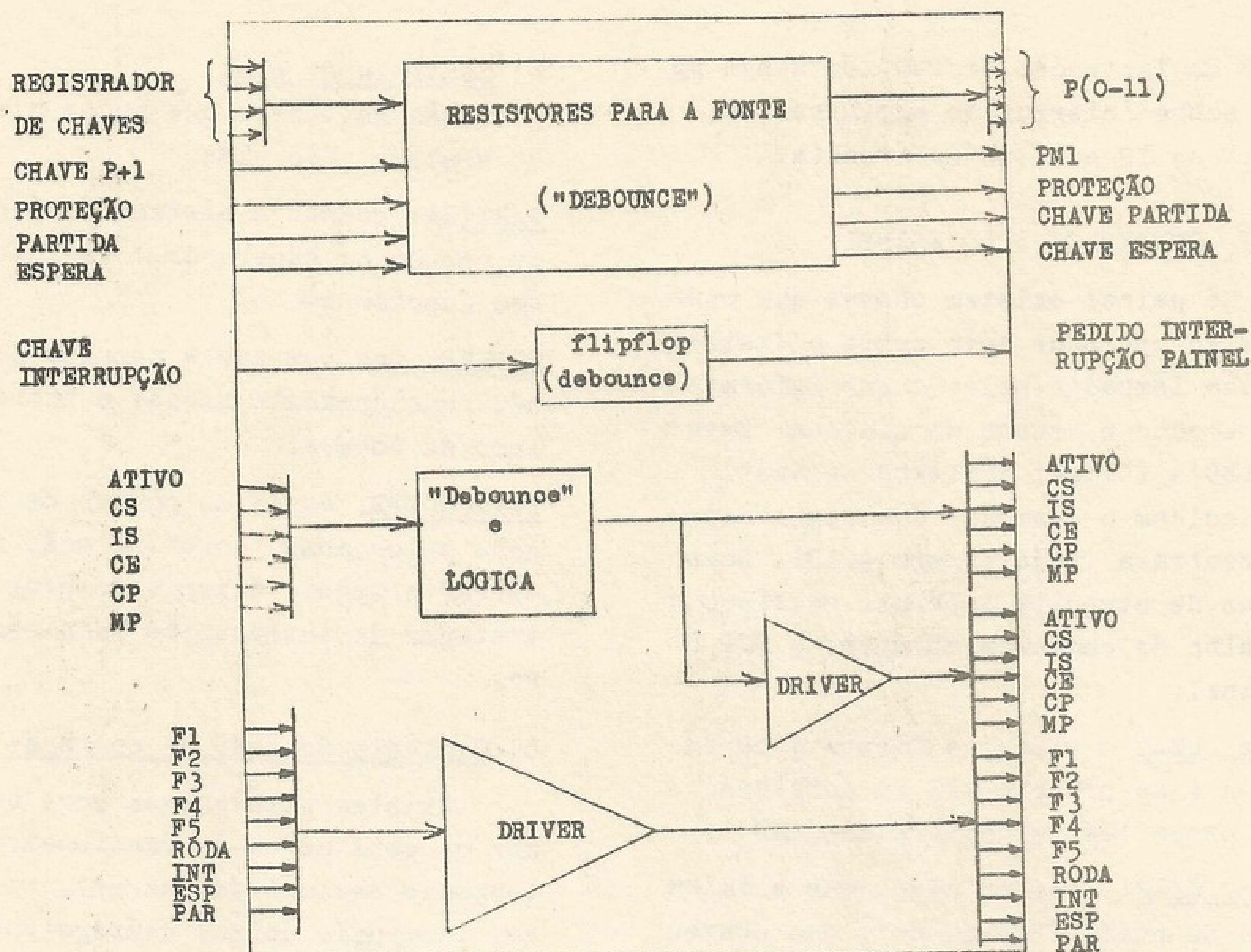



FIGURA 4.20 : Placa do painel (Pilotos e chaves - PCP 2)

carrega endereço (CE), carrega posição (CP) e mostra posição (MP).

O modo normal de operação é aquele onde a sequencialização do programa é da forma comum ou usual. Após a execução de uma instrução, é lida a seguinte e executada, e assim sucessivamente até encontrar uma instrução de PARE ou ESPERE. Os outros modos, cujo funcionamento é diferente desse, são chamados de modos especiais de funcionamento e se classificam em dois grupos:

GRUPO 1:

CICLO ÚNICO: é um modo de funcionamento onde o sistema, a cada acionamento do botão de partida evolui um ciclo apenas. Isso permite que de ciclo em ciclo, o operador do sistema acompanhe a evolução do seu programa. Seus objetivos são principalmente, didáticos e de diagnose de falhas do sistema.

INSTRUÇÃO ÚNICA: é um modo de funcionamento onde o sistema, a cada acionamento do botão de partida executa uma instrução e para no ciclo I da próxima, Serve para o operador acompanhar o seu programa de instrução a instrução. Seu objetivo é o da depuração ("debug") de programas.

GRUPO-2:

CARREGA ENDEREÇO: com o acionamento do interruptor de partida o conteúdo do registrador de chaves do painel é transferido para o registrador de endereço da memória e o contador de instrução.

CARREGA POSIÇÃO: ao se acionar o interruptor de partida, o conteúdo dos 8 bits menos significativos do registrador de chaves do painel é armazenado na posição de memória cujo endereço está no RE.

MOSTRA POSIÇÃO: com o acionamento do interruptor de partida o conteúdo da posição de memória cujo endereço está no RE é transferido para o RD, permitindo ao operador a visualização do conteúdo daquela posição da memória.

A importância dos modos de funcionamento do grupo 2 está na inicialização do sistema e na depuração de programa.

## 4.5 SUMÁRIO

Neste capítulo resumiu-se a descrição do fluxo de dados e da arquitetura do minicomputador. Após o que pode-se iniciar a exposição do projeto da unidade de controle. Na realidade, o projeto da arquitetura e do fluxo de dados correm em paralelo ao projeto da unidade de controle. Apresentou-se a forma final e definitiva da arquitetura e fluxo de dados já que se mostrou impraticável a exposição da evolução.

Logo no início do projeto resolveu-se pelo controle fixo, para evitar-se a memória de controle que não se dispunha. Então, definiu-se a arquitetura levando isso em conta. Foi nesse ponto que a filosofia do controle influiu fundamentalmente na arquitetura: desenvolveu-se um sistema suficientemente simples (na classificação de BELL[39] essa arquitetura é caracterizada como a de um microcomputador típico) para que o controle fixo fosse viável.

# 5 - PROJETO LÓGICO DA UNIDADE DE CONTROLE

## 5.1 INTRODUÇÃO

Na descrição de um projeto de sistemas digitais, muitos autores como Moon(40), Chu(13) e Langdon(44) dividem-no em três aspectos:

- PROJETO FUNCIONAL
- PROJETO SIMBÓLICO
- PROJETO DETALHADO

É uma divisão apenas didática, já que os três aspectos se superpõem, um influindo no outro (fig. 5.1). Para o presente estudo pode-se dizer:

PROJETO FUNCIONAL: definido o porte do sistema, determina-se se a unidade de controle será microprogramada ou não. Define-se então o esquema em bloco da unidade de controle. Esse esquema é analisado (e até simulado) até que se defina em todos os detalhes as características funcionais de cada bloco.

PROJETO SIMBÓLICO: é onde se determinam as cartas de microoperações, tendo-se em vista o fluxo de dados e o repertório de instruções. Aqui também se cuidam dos detalhes especiais como, interrupção de entrada e saída, como se tratar de situações anormais como por exemplo erros de paridade, etc.

PROJETO DETALHADO: é a fase mais extensa, função tanto das definições propostas anteriormente, como também da tecnologia utilizada. É onde se trata dos detalhes no nível de portas lógicas, procurando-se usar técnicas de simplificação, eliminando-se "hazards"(41), cuidando-se dos problemas de atrasos, de "fan-in"(*) e fan-out(**). (41). É onde toda a imperfeição dos dispositivos reais afetará o projeto.

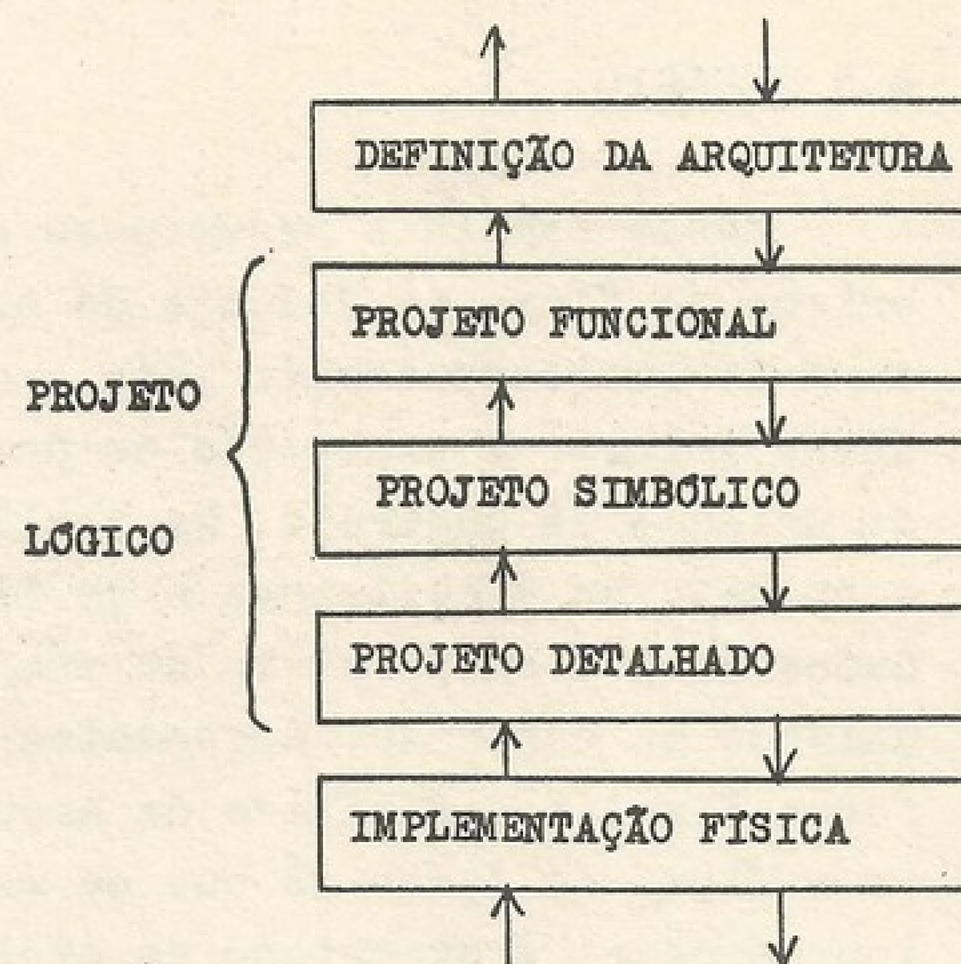



Figura 5.1: Etapas do projeto

Seguindo-se essa linha de exposição do projeto, pode-se dizer que a implementação física segue o projeto lógico. Particionou-se os circuitos em placas de circuito impresso ou "wire-wrap" Prepara-se os "layouts" dos circuitos impressos, as listagens de fiação. Aqui, o engenheiro é influenciado pela tecnologia, pelos padrões de montagem adotados e pela equipe encarregada de montagem mecânica.

Após essas etapas, Langdon(44) inclui uma outra denominada CONFERÊNCIA e DEPURAÇÃO ("CHECK AND DEBUG"). É a fase onde o engenheiro tem que fazer funcionar o protótipo montado.

Para deixar mais claro as etapas do projeto da unidade de controle, preferiu-se expor de uma forma diferente, que dá enfase a certos detalhes, importantes no caso particular de unidade de controle.

---

(*) FAN-IN: número máximo de entradas de um blóco lógico ("gate").

(**) FAN-OUT: número máximo de entradas que podem ser ligadas à saída de um bloco lógico.

Definido o fluxo de dados e a arquitetura, e já que se colocou o registrador de instrução e o contador de instruções no fluxo de dados, não sobram alternativas de esquemas em bloco da unidade de controle que não seja a apresentada na figura 5.2. O decodificador tem como entradas os bits do RI e RE, e suas saídas acionam blocos lógicos do circuito gerador dos sinais de controle A saída do circuito gerador de sinais de controle irá controlar o fluxo de dados. O controle de estado tem a função de sequencializar o programa e fazer o sistema funcionar nos vários modos diferentes como, ciclo único, carrega endereço, etc. O relógio central fornece as referências de tempo para o sistema.

O projeto do decodificador e do relógio central é o que menos oferece dificuldades, dada a sua simplicidade. A complexidade aumenta muito no projeto do controle de estado, um circuito sequencial. Pode-se dizer que toda característica sequencial do computador é ditada por esse circuito. O circuito gerador dos sinais de controle é a parte mais característica da unidade de controle. Seu projeto é feito a partir das cartas de microoperações. É um circuito combinatório com grande número de saídas ( aproximadamente 50 ) e entradas ( aproximadamente 90 ) e por isso exige o desenvolvimento de técnicas especiais de projeto.

A figura 5.3 ilustra, em um diagrama, a exposição que se seguirá da unidade de controle.

Aqui existe a mesma dificuldade anteriormente citada: apesar de se expor as partes isoladamente, deve-se frizar que todas correm em paralelo, num processo interativo, cada uma influindo nas outras.

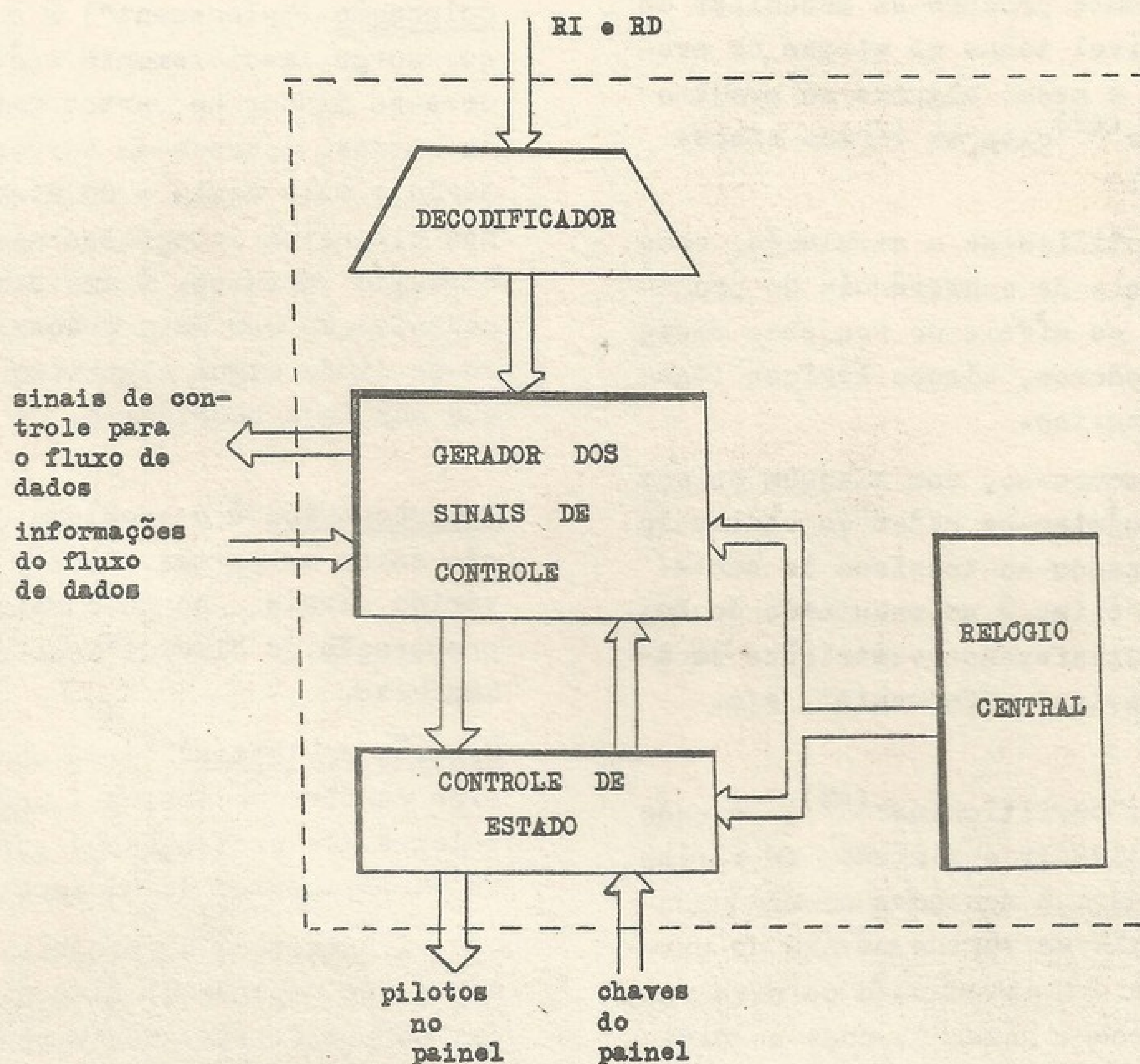



FIGURA 5.2 : UNIDADE DE CONTROLE

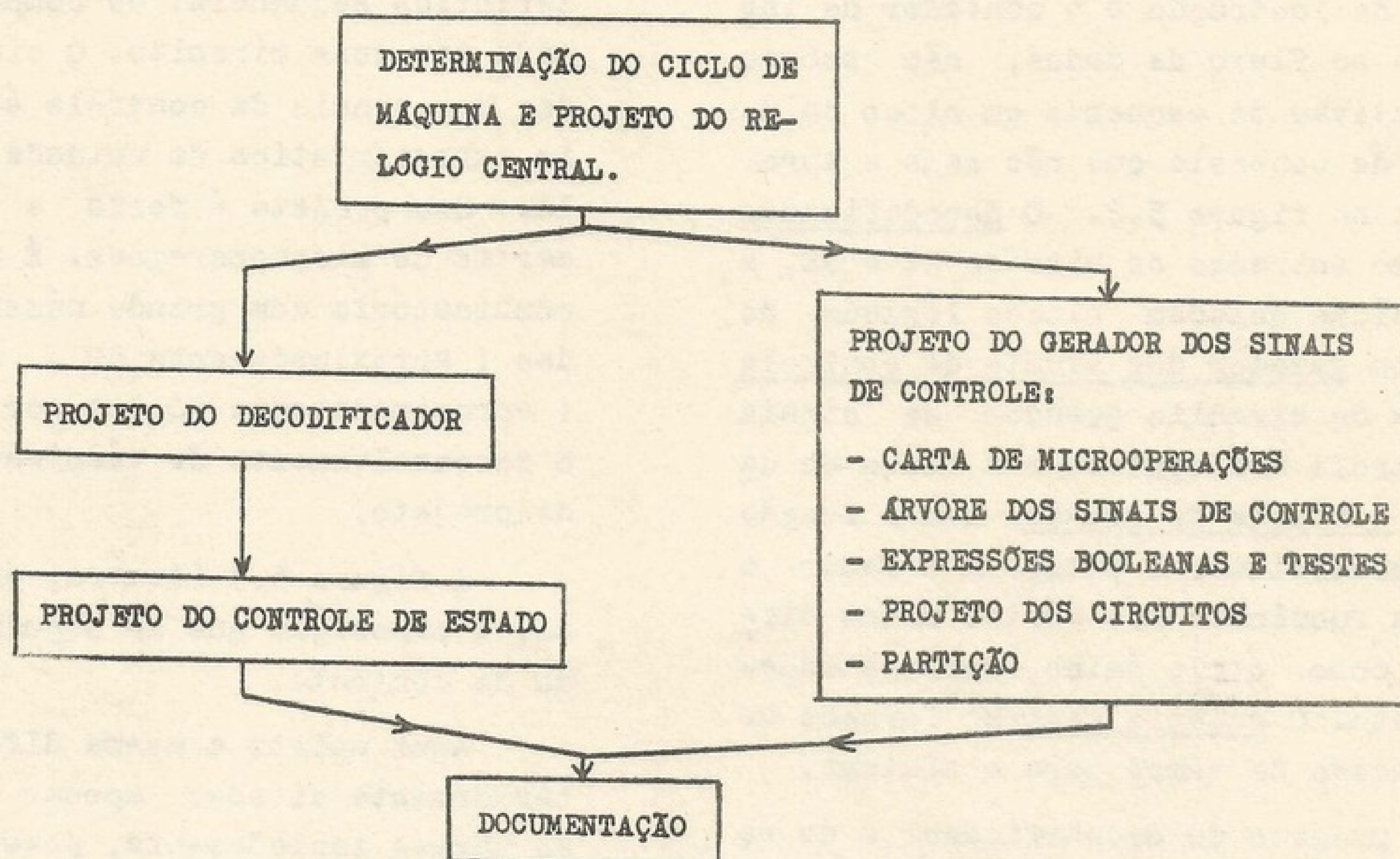



Figura 5.3 : Itens do projeto lógico da UNIDADE DE CONTROLE

Atualmente procura-se mecanizar ao máximo possível todas as etapas do projeto lógico e mesmo algumas do projeto físico. Case[42] cita as várias etapas mecanizadas:

Simulação: utiliza-se a simulação, como uma ferramenta de conferência de projeto em todos os níveis do projeto: sistema, registradores, blocos lógicos ("gates") e circuitos.

Síntese: procura-se, com a ajuda do computador, projetar-se redes de blocos lógicos utilizando-se técnicas de síntese[45] cujo ótimo é um resultado de baixo custo, satisfazendo restrições de atrasos, "fan-ins", "fan-outs", etc.

Partição: ("partitioning"[46]) é onde se procura dividir o sistema em várias partes que seriam montadas separadamente. Aqui, existem vários níveis de partição, sendo o mais comum o do nível de blocos lógicos ("gates"), onde se divide um circuito em placas de circuitos impressos.

Colocação ("placement") é o problema que surge imediatamente após a partição; deve-se dispor as partes umas relativas as outras. Aparece em vários níveis, sendo o mais comum o da distribuição dos circuitos integrados na placa de circuito impresso. É uma das partes de mecanização das mais árduas e desconhece-se ainda algum algoritmo eficiente que não seja heurístico.

Interconexão: é o problema da interligação entre as partes. Também existe nos vários níveis, sendo o mais comum o da preparação do "layout" de um circuito impresso.

Geração de testes[47]: é onde se determina padrões de testes ( combinações de valores das excitações ) para se testar as várias partes do sistema.

A automação do projeto é uma ferramenta que depende da disponibilidade local do projetista. Neste projeto, automatizou-se algumas partes, possíveis dentro dos recursos materiais do Laboratório.

## 5.2 DETERMINAÇÃO DO CICLO DA MÁQUINA E PROJETO DO RELÓGIO CENTRAL

Ciclo da máquina é um conceito sem uma definição precisa. Alguns o associam ao ciclo da memória principal, outros, principalmente quando a máquina é microprogramada, com o ciclo da memória de controle. Neste texto assumir-se-à o mesmo conceito que Chu[13] adotou: " ... ciclo da máquina é, usualmente estabelecido em relação à memória... a cada acesso à memória corresponde a um ciclo de máquina". E, considerando que a memória principal é o elemento mais lento do fluxo de dados, esse conceito fica coerente com as palavras de Langdon[44] "o pior caso de atraso encontrado no fluxo de dados para uma microoperação não deve exceder o tempo atribuído ao ciclo da máquina".

Como já se disse, a memória Philips FI 21[35] tem um ciclo completo de 1.6 µseg. Lembrando-se de ciclo partido ("split cycle"), onde utiliza-se dos ciclos ler e escrever separadamente, é razoável pensar-se num ciclo de máquina maior que 1.6 µseg, para que se disponha de algum tempo entre os dois modos, ler e escrever. Definiu-se então o ciclo da máquina de 2 µseg lembrando também que precisa-se de algum tempo depois que a memória terminou um ciclo para atualizar o sistema, antes que ela entre em um novo ciclo. Dentro desse tempo de 2 µseg pode-se encaixar os ciclos de memória que são de três tipos: a) Modo 4, ler e restaurar (ciclo completo), b) Modo 1, ler somente (meio ciclo) e c) Modo 3, escrever somente (meio ciclo).

As microoperações serão encaixadas também, dentro do ciclo da máquina. Depois de algumas tentativas de preparação da carta de microoperações pode-se ver que era conveniente definir-se 8 segmentos de tempo. Ou seja dividir-se o ciclo da máquina em 8 partes, que seriam o menor tempo reconhecido pelo sistema. Uma razão que conduziu a escolha dos segmentos de tempo de 250 µseg foi a tolerância de projeto. Como se utilizou de uma memória lenta comparada aos circuitos, sua divisão em 8 partes resultou em tempos grandes (da ordem de 15 atrasos de blocos lógicos). Isso ofereceu uma vantagem de projeto pois ampliou as tolerâncias. Ware[19] afirma que "podem existir tolerâncias associadas aos intervalos de tempo e atrasos, os segmentos de tempo ("clock interval") refletirão tais problemas de tolerância".

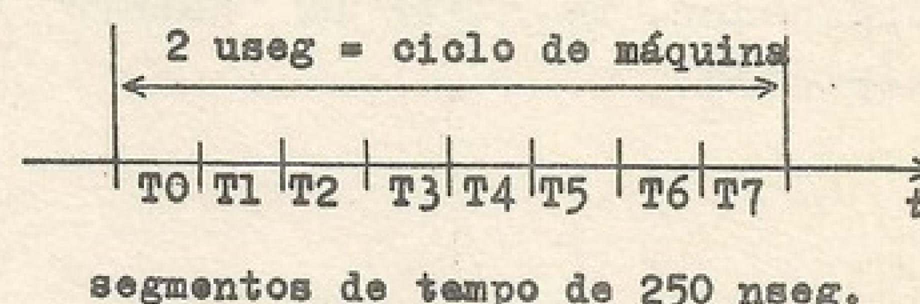



**Figura 5.4: Divisão do ciclo de máquina**

Como se vê à frente, a determinação do ciclo da máquina é feita, durante a preparação da carta de microoperação. Na figura 5.4 apresenta-se o ciclo da máquina com seus segmentos de tempo: T0, T1,...T7.

Então, uma microoperação, de duração de 1 segmento de tempo, poderá ser locada em 8 diferentes posições dentro do ciclo da máquina. O circuito que fornecerá os sinais elétricos das microoperações (gerador dos sinais de controle) receberá sinais de referência de tempo de um circuito chamado RELÓGIO CENTRAL.

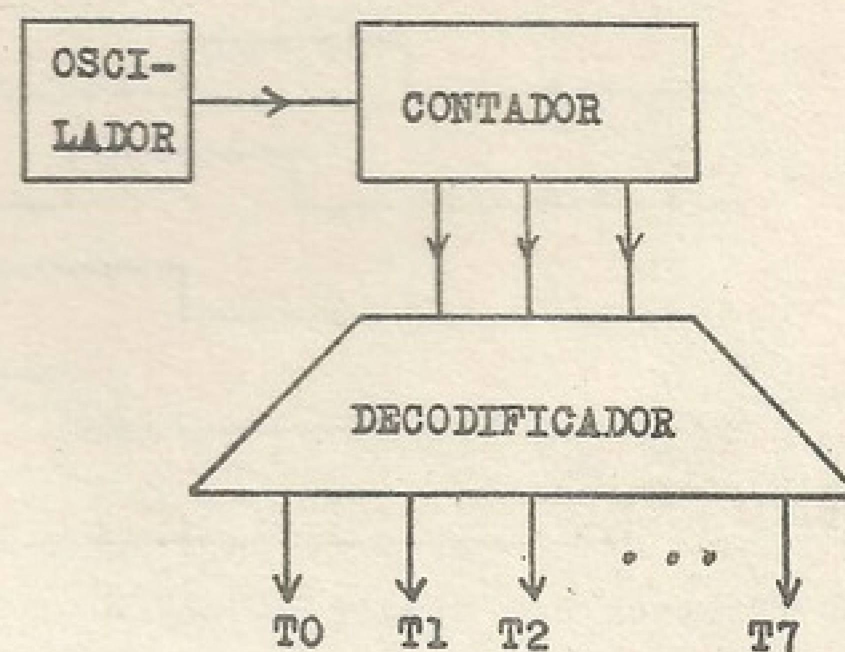
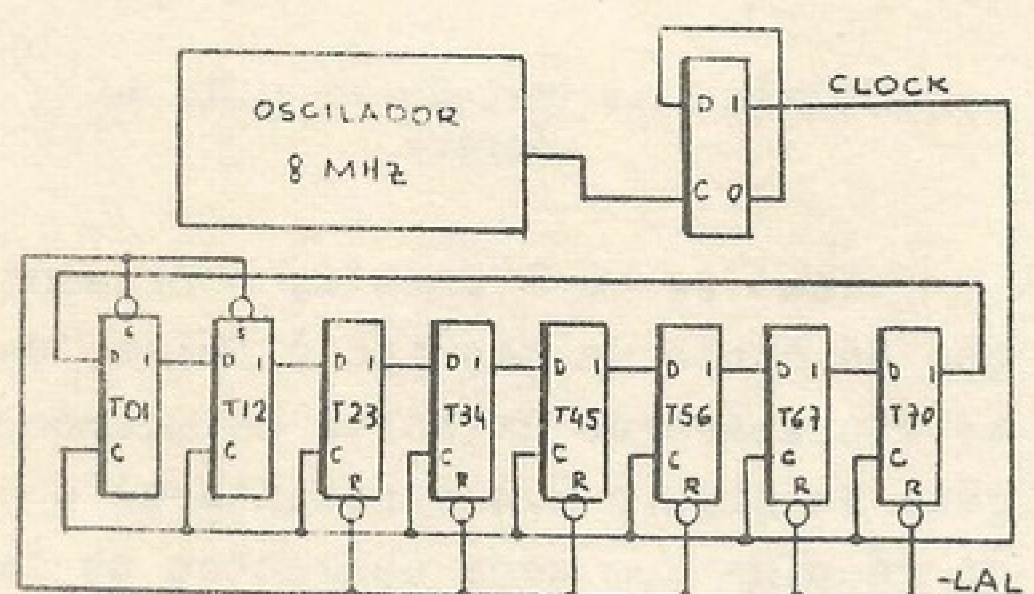


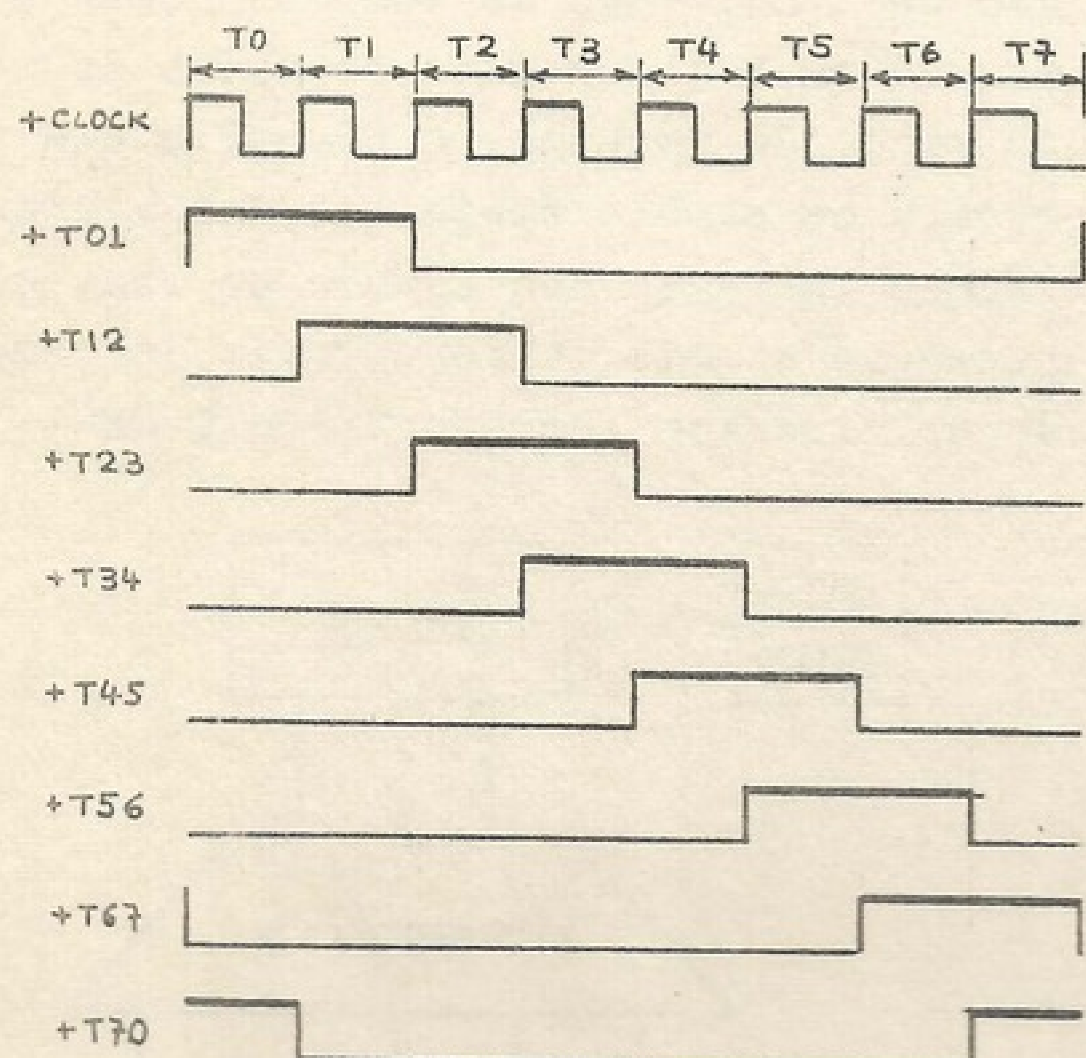

**Figura 5.5: Esquema de um relógio central**

O relógio central é, basicamente, constituído de três partes: a) um oscilador a cristal, b) um contador e c) um decodificador. Veja figura 5.5.

Essa estrutura tem vários inconvenientes: o primeiro é o fato do contador precisar ser no código de Gray[10] para evitar "hazards", ou seja para evitar que na transição do contador exista algum sinal de tempo incorreto na saída do decodificador. Outro inconveniente surge, quando se quiser um sinal de tempo composto e se realizar um "or" de vários segmentos de tempo da saída do decodificador. Existirá um "hazard" combinatório[41].




a) esquema simplificado dos circuitos




b) gráficos dos sinais de saída

FIGURA 5.6 : RELÓGIO CENTRAL

Por esses motivos optou-se pela estrutura vista na figura 5.6a. É a técnica de utilização do "overlapped ring counter"[43]. É um circuito deslocador com giro ("shift rotate"), inicializado com o código 11000000. Esse código girando, cada flip flop gerará um sinal de tempo duplo, e as formas de onda geradas por essa estrutura pode ser vista na figura 5.6b. Dessa maneira pode-se obter sinais de tempo simples com um "AND" de dois tempos duplos ( exemplo: T4 = T34.T45), e sinais de tempos longos são obtidos com "OR" sem possibilidade de "hazards" combinatório (exemplo: T456 = T45 + T56).

A inicialização (11000000) citada é feita com o sinal LAL (limpa ao ligar) que é um pulso gerado quando se liga o sistema.

O relógio central foi montado em uma placa de circuito impresso, conforme visto na prancha IX. Ali, pode-se notar que o relógio central gera tempos simples e duplos. Alguns deles são gerados em paralelo (duas vezes) por problemas de "fan-out". O oscilador é um colpitts modificado.

## 5.3 PROJETO DO DECODIFICADOR

O agrupamento das instruções em conjuntos que tem alguma característica comum, é uma tarefa importante para futuras simplificações. Esse agrupamento deve ser feito antes da definição dos códigos de cada instrução para que o decodificador possa ser projetado, sem muita complexidade, para gerar os sinais de grupos também. Por exemplo se as instruções CAR (carrega acumulador) e SOM (soma no acumulador) pertecem ao mesmo grupo, a presença de uma delas, além de ativar a sua específica saída do decodificador, ligará também a saída do decodificador correspondente ao grupo.

Na tabela 5.1 encontra-se a relação dos grupos e sub grupos de computador em questão.

O decodificador projetado, seguindo o esquema de entradas-saídas mostrado na figura 5.7, tem saídas indicando todas as instruções e também os sinais de grupos e subgrupos indicados na tabela 5.1.

TABELA 5.1

AGRUPAMENTO DAS INSTRUÇÕES

| GRUPOS | SUBGRUPOS | INSTRUÇÕES |
|---|---|---|
| CASP | | CAR, ARM, SOM, PUL |
| | CAC+SOM | CAR, SOM |
| | ARA+PLI | ARM, PUL |
| SUS+PUG | | SUS, PUG |
| PAN+PAZ | | PLAN, PLAZ |
| IME | | SOMI, NAND, XOR, CARI, DESLOCAMENTOS |
| SAE | | SAI, ENT, SAL, FNC |
| MIC | | (todas micros) |
| | MICG1 | |
| | MICG2A | |
| | MICG2B | |

Como será frizado à frente, permitiu-se um atraso de decodificação de 200 nseg, o que significa que se pode projetar um decodificador de até 10 níveis lógicos da família TTL - 7400.

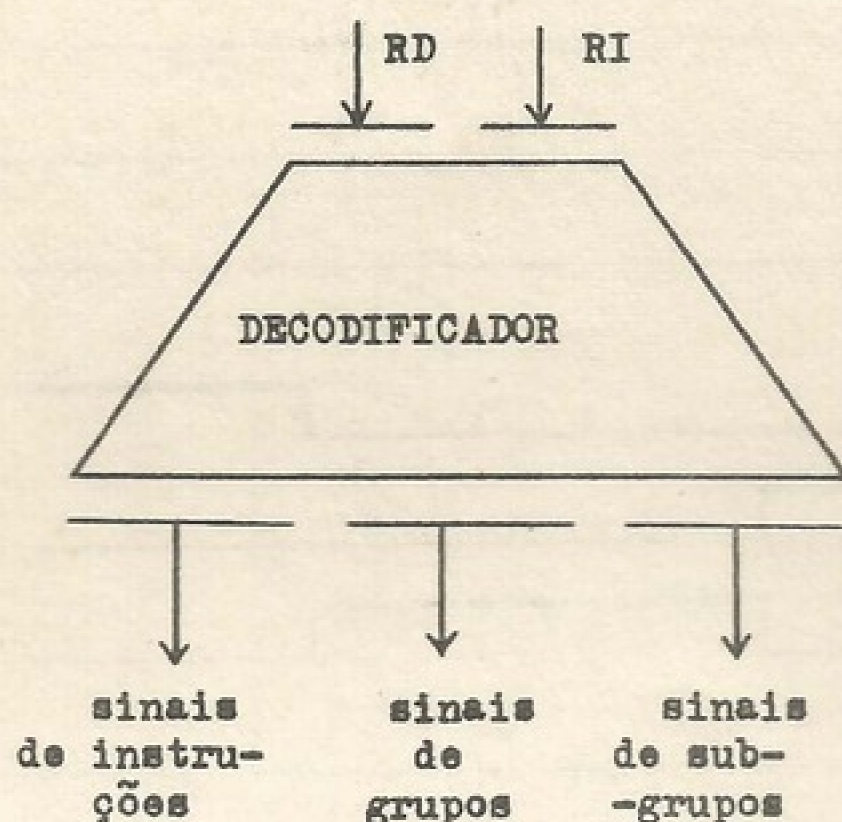



Figura 5.7: Modelo de entrada/saída do decodificador

Então, o projeto do decodificador ficou sem restrições de atraso, devendo apenas satisfazer a condições de "fan-out". Para que a sua entrada não "carregasse" muito as saídas dos registradores RD e RI, colocou-se inversores na sua entrada. Dada a grande quantidade de entradas e saídas desse circuito combinatório, torna-se impraticável qualquer tentativa de aplicação das técnicas convencionais de minimização. Por isso o seu projeto foi realizado heuristicamente, num processo iterativo de tentativas e mudanças.

Na prancha X encontra-se dos detalhes do circuito do decodificador implementado.

## 5.4 PROJETO DO CONTROLE DE ESTADO

Durante o funcionamento normal, pode-se distinguir 5 tipos diferentes de ciclo de máquina. São chamados de FASE 1 FASE 2, FASE 3, FASE 4 e FASE 5. As fases 1 e 2 são ciclos de busca ou ciclos I. Na fase 1 a primeira palavra da instrução é lida na memória e, na fase 2 a segunda palavra da instrução. A fase 3 é o ciclo de execução onde é lido ou gravado o operando na memória, e, como algumas instruções exigem dois acessos à memória para serem executadas, existe a FASE 4. Instruções indexadas exigem um ciclo de máquina diferente, para ler o índice na posição zero da memória e calcular o endereço efetivo. É a fase 5.

Resumindo

Fase 1, Fase 2 } fases de "fetch" ou busca

Fase 3, Fase 4 } fases de execução

Fase 5 } fase de cálculo de endereço efetivo

Durante a execução de um programa a UCP vai pulando de fase em fase, sob o comando da unidade de controle. Por exemplo, da fase 1 de uma instrução de SOMA INDEXADA evolui para a fase 2 daí para a fase 5 e volta para a fase 3, para depois entrar na fase 1 da instrução seguinte.

Note-se que o simples controle das fases resolve o problema da sequencialização de programa. Se em toda a fase 1 ler-se na memória a primeira palavra de uma instrução, endereçada pelo contador de instruções, e se em toda fase de busca atualizar-se o contador de instruções, basta, após executar-se cada instrução, encaminhar-se, automaticamente, para a fase 1 para que a sequencialização do programa seja garantida.

Para o controle das fases o "gerador de sinais de controle" fornece cinco sinais, mutuamente exclusivos:

CF1 - condição para FASE 1

CF2 - condição para FASE 2

CF3 - condição para FASE 3

CF4 - condição para FASE 4

CF5 - condição para FASE 5

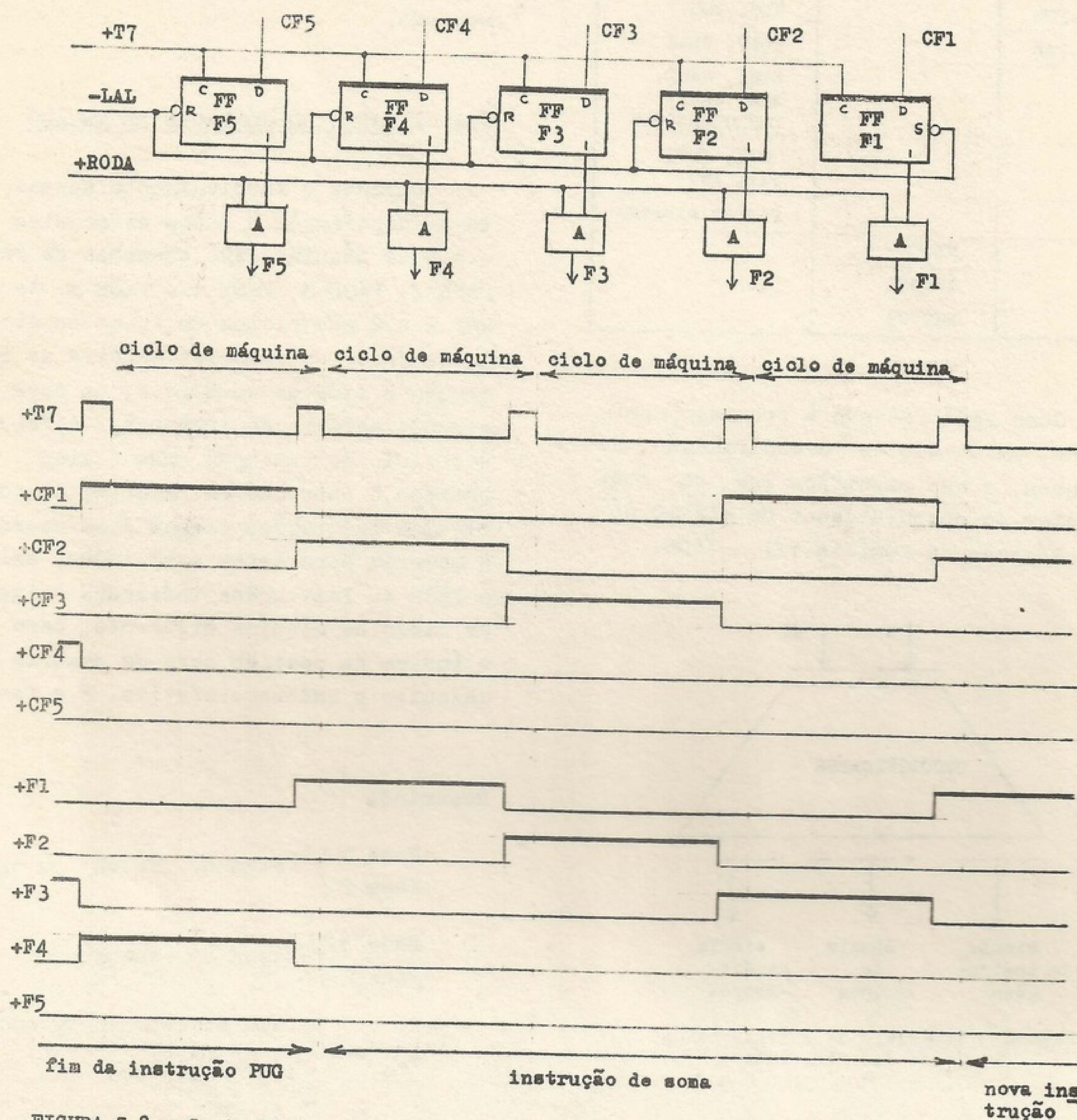



FIGURA 5.8 : Controle das fases

Durante uma fase esses sinais indicam qual será a fase seguinte. Por exemplo, se se está na Fase 2 de uma instrução de CAR, CF3 está no nível "1" e os outros no nível "zero", indicando que a fase seguinte é a fase 3.

Na placa de controle de estado existem 5 flip flops cujos estados indicam fases, um flip flop para cada fase. O controle das fases é feito colocando-se os sinais "condições de fase" na entrada D dos flip flops e no início de T7 dando-se um pulso na entrada de controle (clock) do flip flop. A figura 5.8 ilustra essa idéia.

Durante o funcionamento, o sistema vai pulando de fase em fase.seguindo o que lhe dita os cinco sinais de condições de fase. Se por exemplo, após uma instrução de PUG (fase 4 é a última fase dessa instrução), o sistema encontrar uma instrução de SOMA,ocorrerá o seguinte: durante a fase 1 o sinal CF2 estará no nível 1, o que provocará, em T7, o disparo do flip flop "fase 2"; na fase 2 será CF3 quem será "1" fazendo com que o flip flop "fase3" se ligue em T7; na fase 3, CF1 será ativado para ler a nova instrução. Na figura 5.8 tem um diagrama que ilustra, no tempo, essa evolução.

Ao ligar o sistema, deve-se inicializá-lo na fase 1. O sinal "-LAL", visto na figura 5.8 tem essa função.

Com essa filosofia de fases, fica simples parar o sistema no meio do processamento (instrução de pare, por exemplo). Se o"gerador dos sinais de controle" utilizar sempre o sinal de fase para produzir qualquer sinal, a inibição dos cinco sianis de fase, inibirá todo o sistema. O sinal RODA (figura 5.8) quando igual a "ZERO" inibe as fases parando o computador.

É controlando o sinal de Roda que se opera nos modos especiais de funcionamento, como o ciclo único e instrução única. Esse sinal é composto por três outros (figura 5.9): RODA 1, RODA 2 e RODA 3:

RODA 1: usado durante o modo normal de funcionamento. Fica sempre no nível "1" até que se pare o sistema. As condições para ligá-lo e desligá-lo são as seguintes:
LIGAR: quando aciona-se o botão de partida ou quando se está no estado de espera e chega um pedido de interrupção.
DESLIGAR: sempre antes de uma FASE 1 nas seguintes circunstâncias: instrução de PARE, instrução ou acionamento do ESPERE ou tirando-se do modo normal de funcionamento.

RODA 2: usado no modo especial instrução única (IS). É ligado quando se está nesse modo de funcionamento e se pressiona o interruptor de partida. É desligado sempre que se for entrar numa fase 1.

RODA 3: utilizado no modo especial ciclo único (CS). É um sinal que fica no nível 1 por apenas um ciclo. Utiliza-se o próprio sinal de partida, que dura um ciclo, para gerá-lo.

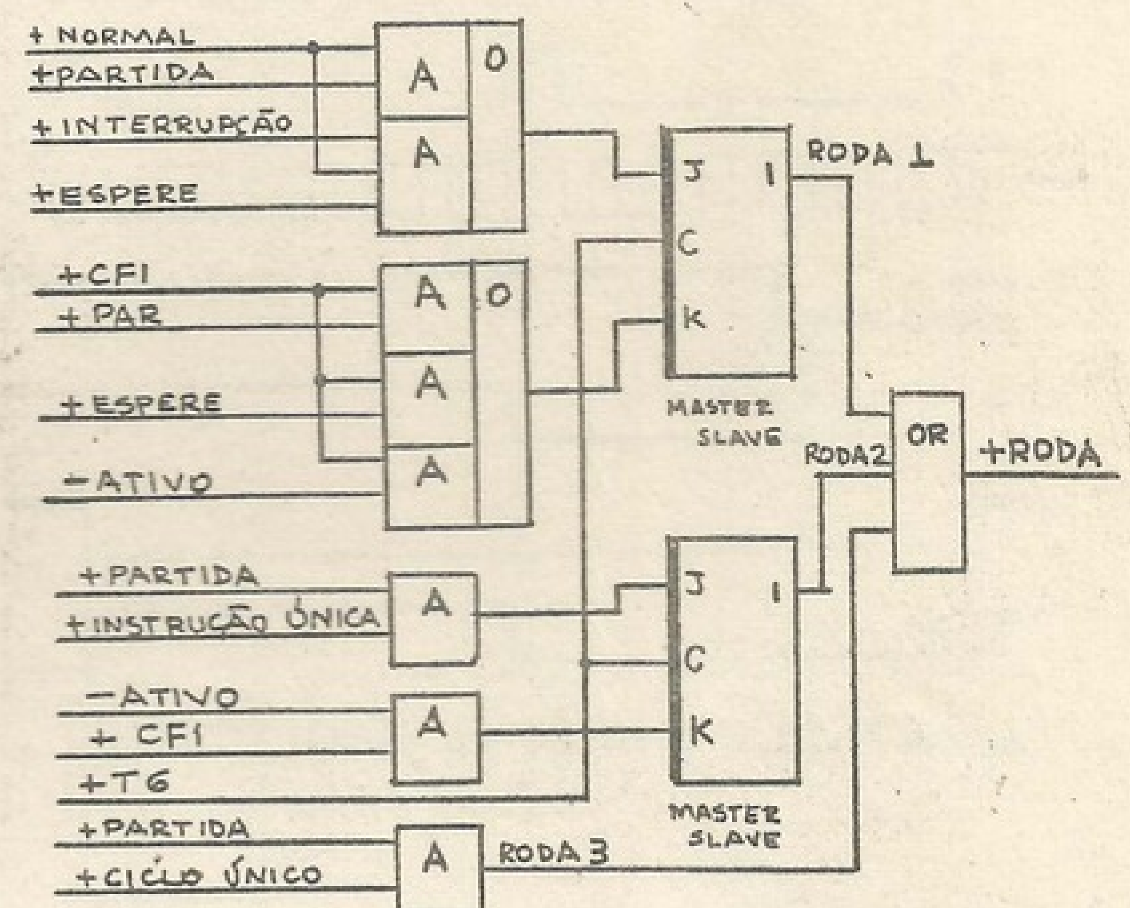



**Figura 5.9 : geração do sinal RODA**

Uma tarefa importante do circuito "controle de estado" é a provisão da interface entre o operador e o sistema. Já se comentou que o operador, ao selecionar o modo de operação, acaba, no final, controlando o sinal de RODA. O acionamento do interruptor de ESPERE irá ligar o flip flop correspondente, que, por sua vez, desligará o sinal RODA 1 . O interruptor de PREPARAÇÃO ("PRESET") age diretamente no circuito de geração do LAL (limpa ao ligar), parando o relógio central, forçando fase 1, desligando interrupção, etc.

Um pouco mais delicada é a geração do sinal de partida, já que a cada pressionada da chave ele deve existir por apenas um ciclo, começando em T0 e acabando em T7. Isso porque esse sinal é usado em inúmeros pontos onde o sincronismo é vital. Um circuito com três flip flops, sendo o primeiro eliminador de ruído ("bounce") da chave gera esse sinal, chamado PARTIDA.

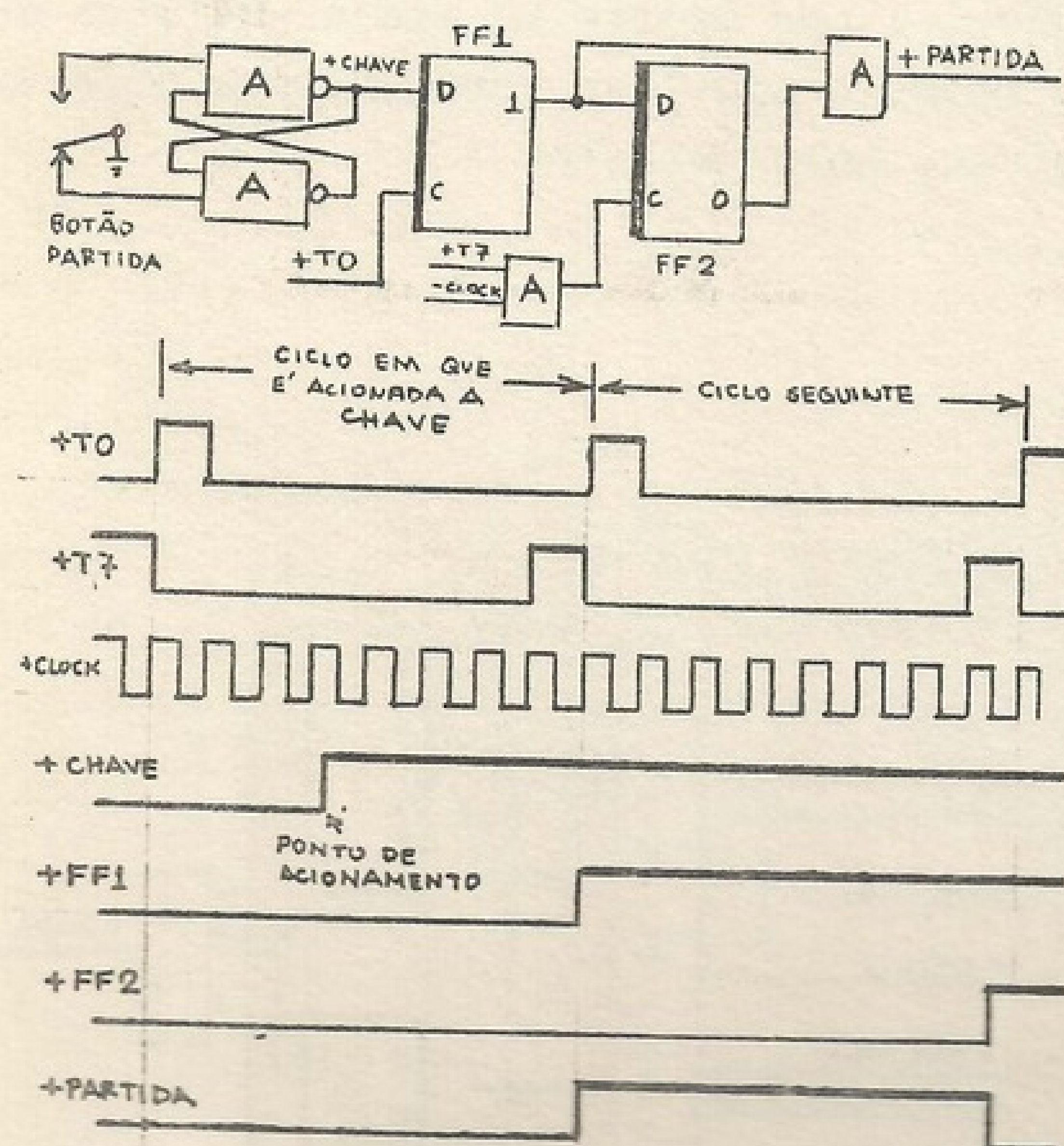
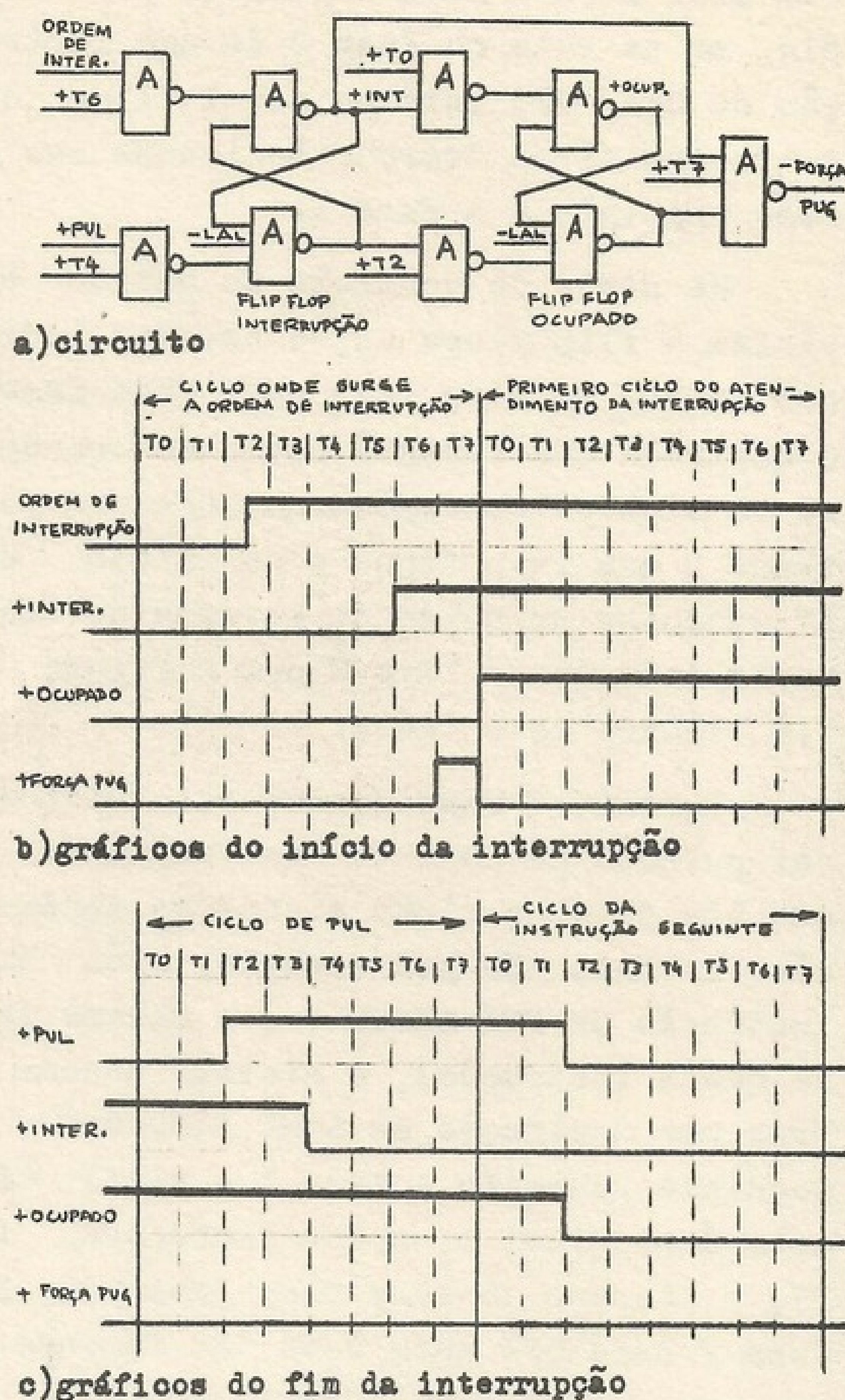



Figura 5.10 : Geração do sinal PARTIDA

Outra função do circuito "controle de estado" é controlar a interrupção. Quando chega um pedido de interrupção ele é atendido antes que a próxima fa-




a) circuito

b) gráficos do início da interrupção

c) gráficos do fim da interrupção

FIGURA 5.11 : MECANISMO DA INTERRUPÇÃO

se 1 se inicie. O pedido de interrupção passa por algumas portas que testam certas condições (ver se não está ocupado, se não está inibido, se é fase 1, etc ) e se transforma num sinal chamado "ordem de interrupção". Esse sinal (figura 5.11) dispara um flip flop chamado "interrupção" no tempo T6. No segmento de tempo T0 seguinte é disparado o flip flop "ocupado". Nesse meio tempo (em T7) é gerado o sinal FORÇA PUG que realiza o seguinte:

Liga Fase 3

Desliga Fase 1

Força instrução de PUG no RI
("set" RI)

Força endereço 2 no RE

Isso tudo equivale a iniciar uma instrução de PUG para a posição 2 da memória.

Os flip flops INT e OCUPADO ficam ligados durante todo o tempo de atendimento da interrupção, e apenas uma instrução de PUL (ou acionamento da chave "preparação") os desliga. Para que não se perca o endereço de volta ao programa principal, que durante o atendimento da interrupção ficou armazenado na posição 2 da memória, o flip flop OCUPADO é desligado no ciclo seguinte ao ciclo de PUL, aproveitando-se o fato de que enquanto o flip flop ocupado se mantiver no nível "1" nenhum outro pedido de interrupção será atendido.

Na prancha XI existem os circuitos desta placa. Ali se encontram alguns outros detalhes que merecem algumas palavras:

Gerador do LAL: é um circuito de componentes discretos. Basicamente é um SCR que, ao ligar o sistema se encontra cortado, fazendo com que LAL=1. Ao se acionar, pela primeira vez, o interruptor de partida o SCR conduz, levando o LAL para zero.

Sinal FN: é um sinal, chamado FASE NORMAL, controlado por uma chave que se encontra dentro da máquina. Quando FN="1" o sistema funciona normalmente. Ao se acionar a chave fazendo FN="0", deixa de existir a evolução das fases, ficando o sistema rodando preso em uma fase. É utilizado na depuração do sistema.

Sinal PARTIDA ATRASADA: é um sinal que inicia no T6 de um sinal de partida e acaba no T5 do ciclo seguinte. Ele é usado para limpar os flip flops PARE e ESPERE. Quando se aciona o interruptor de partida para se sair de um desses estados, o sinal de RODA 1 é energizado no final do ciclo de partida. Com o RODA 1 ligado, nova instrução será lida e o decodificador que estava dando o sinal da instrução anterior de PARE ou ESPERE passa a indicar a nova instrução. Então pode-se limpar (no ciclo seguinte ao sinal de partida) o estado de PARE ou ESPERE.

Resumindo

O circuito controle de estado tem por função controlar as fases, e com a correta sequencialização delas conseguir a sequencialização do programa. Provê o atendimento das interrupções sincronizando-as ao sistema. Opera como interface com as chaves do painel preparando seus sinais. Um sinal chamado RODA inibe as fases quando se quiser parar o sistema.

## 5.5 DEFINIÇÕES DO PROJETO DO GERADOR DOS SINAIS DE CONTROLE

Esta é a parte do projeto que tem toda a característica de unidade de controle. É um circuito combinatório que recebe como entradas a saída do decodificador, os sinais de tempo, os sinais de fase, de modo de operação e sinais do fluxo de dados. Como saída tem os sinais de controle do fluxo de dados para executar a instrução corrente e para sequencializar o programa.

No seu projeto deve-se levar em conta todas as instruções que serão executadas em todos os detalhes. Deve-se conhecer o fluxo de dados, de modo a se poder definir todos os atrasos dos circuitos e vias. Ware[19] afirma que: "Para cada instrução é possível listar cada evento (microinstrução) que precisa ocorrer e também a sequência em que eventos precisam ocorrer. A princípio nós precisamos ou calcular ou estimar o tempo requerido para que cada evento se complete. Nós podemos então construir a carta de microoperações ("timing chart")..."

Construída a carta de microoperações, o passo seguinte é implementá-la em circuitos. É uma tarefa árdua dadas as dimensões e a quantidade de variáveis que torna ineficiente todas as técnicas de síntese. É neste ponto que os autores se calam: como a partir da car-

ta de microoperações obter-se o circuito. Considerando-se ainda que existem restrições de tempo, alguns sinais devem ser gerados com um mínimo de atraso.

## 5.6 DETERMINAÇÃO DA CARTA DE MICROOPERAÇÕES

Como já se disse, é na definição da carta de microoperações que se define o ciclo da máquina.

Esta é a fase do projeto onde o projetista fica mais perto da definição da arquitetura. Sua função aqui, é analisar cada instrução, e detalhá-la microinstrução a microinstrução, distribuindo-as dentro do ciclo da máquina. Tudo o que o sistema executará em cada instrução é determinado nesta fase.Aqui existe uma enorme interação entre os projetistas do controle e do fluxo de dados. É onde o controle pode influir muito na arquitetura.

Pode-se dividir esta etapa nas seguintes partes:

1) Relação de todas as instruções, cada uma com sua lista de microoperações corretamente distribuídas.
2) Caracterização das fases. Sempre que possível, é conveniente reservar 1 ciclo de máquina para cada ciclo de memória.
3) Construção da carta de microoperações, fase a fase, procurando agrupar as instruções cujas fases são idênticas.
4) Diminuição do ciclo de máquina e redução do número de segmentos de tempo.
5) Relocação das microoperações dentro do ciclo da máquina para aumentar a uniformidade.

Ao se executar esse trabalho deve-se ter em mente duas grandezas: custo do circuito e velocidade. Ao se fazer cada fase corresponder a um ciclo de memória, procura-se ganhar velocidade. Quando não se tem acesso à memória, evita-se gastar um ciclo de máquina. Por exemplo as instruções de entrada e saída devem ser executadas durante as fases de busca, já que elas não utilizam a memória. Para ganhar velocidade também deve-se tentar diminuir o ciclo da máquina. Podendo-se comprimir mais as microoperações e, se possível fazê-las simultâneas, deve-se então redistribui-las de modo a tornar menor o ciclo da máquina. O agrupamento das instruções idênticas diminue o custo, já que elas utilizarão os mesmos circuitos. Ao se relocar as microinstruções de modo a uniformizar a distribuição também diminue o custo. Exemplo: existem algumas instruções que na fase 3 carrega o acumulador com alguma coisa (instruções de SOMA, CAR ou MICROS). Essas microoperações devem ser feitas nos mesmos segmentos de tempo. Com isso economiza-se em circuitos. O projetista deve ter em mente que, as vezes, a inclusão de uma microinstrução irrelevante pode diminuir o custo. Veja um exemplo: na maioria das instruções, na fase 2 incrementa-se 1 ao contador de instruções, com exceção do PULO, onde o CI é carregado com o campo de endereço da instrução. A unidade de controle que soma 1 ao CI em todas as instruções com exceção da de PULO é mais cara que aquela que soma sempre, sem ter que distinguir o PULO.

Nas unidades de controle microprogramadas essa é a tarefa do microprogramador.

Nas figuras 5.12, 5.13 e 5.14 encontram-se as cartas de microoperações desta unidade de controle. Ali, é importante observar que nos tempos T0, T1 e T2 da fase 1 todas as instruções são iguais, e já que o acesso à memória é feito no final de T1, pode-se ter um atraso de decodificação de até 250 nseg (T2).

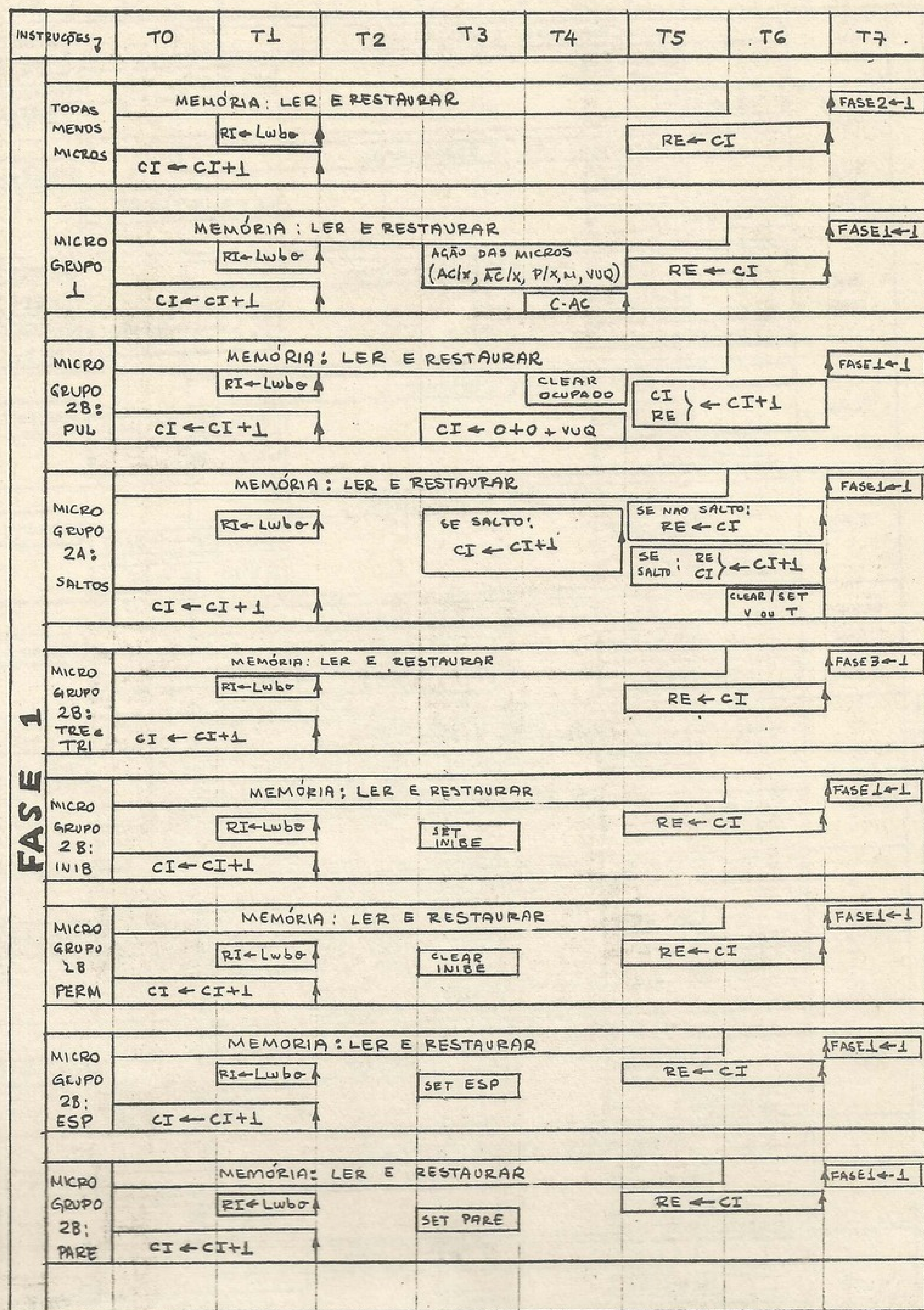



FIGURA 5.12: CARTA DE MICROOPERAÇÕES DA FASE 1

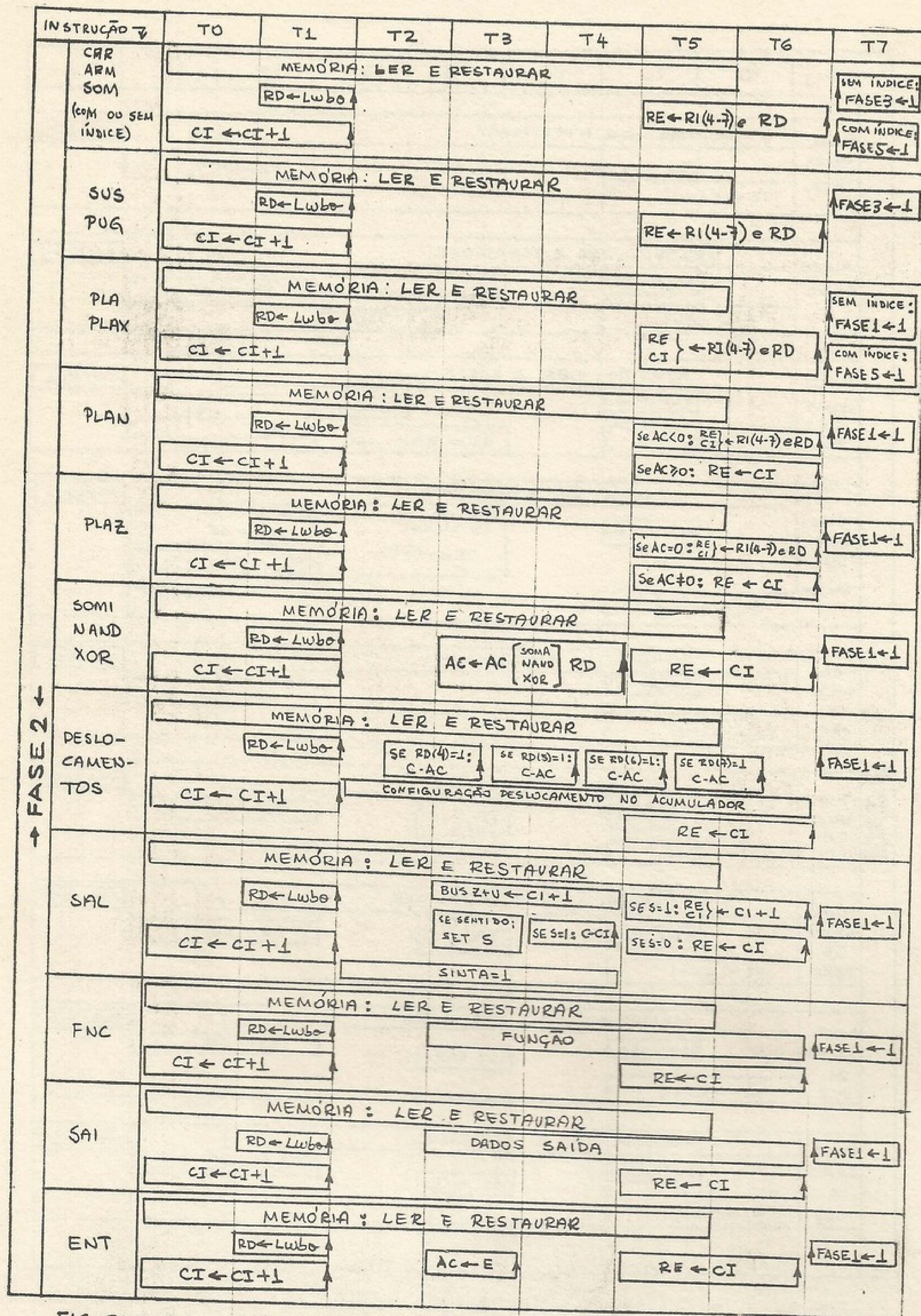


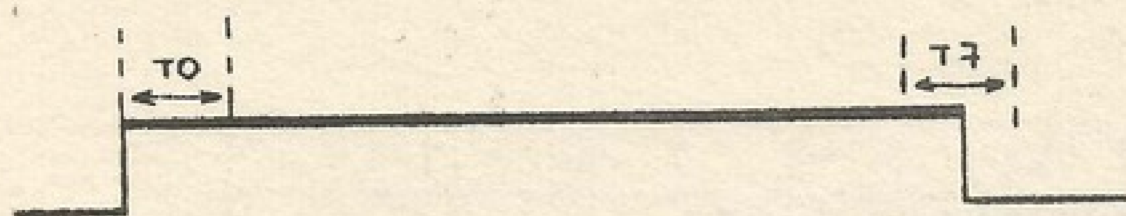

FIGURA 5.13: CARTA DE MICROOPERAÇÕES DA FASE 2

| | INSTRUÇÕES | T0 | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|---|---|
| FASE 3 | CAR<br>CARX | MEMÓRIA : LER E RESTAURAR | RD ← Lwbo | | AC ← RD | | RE ← CI | | FASE1 ← 1 |
| | ARM<br>ARMX | MEMÓRIA: LER SOMENTE | RD ← AC | | MEMÓRIA : ESCREVER | | RE ← CI | | FASE1 ← 1 |
| | SOM<br>SOMX | MEMÓRIA : LER E RESTAURAR | RD ← Lwbo | | AC ← AC + RD | | RE ← CI | | FASE1 ← 1 |
| | PUG | MEMÓRIA: LER SOMENTE | RD ← CI(0-3) | | MEMÓRIA: ESCREVER | | RE ← RE+1 | | FASE4 ← 1 |
| | TRE | MEMÓRIA: LER SOMENTE<br>1/Lwa | RD ← Lwbo | | MEMÓRIA : ESCREVER<br>RE ← RD | | | | FASE4 ← 1 |
| | TRI | MEMÓRIA: LER SOMENTE<br>0/Lwa | RD ← Lwbo | | MEMÓRIA : ESCREVER<br>RE ← RD | | | | FASE4 ← 1 |
| | SUS | MEMÓRIA: LER SOMENTE | RD ← Lwbo | se ZDZ=0: RD ← RD-1<br>se ZDZ=1: SET S | se S=1: CI ← CI+1 | MEMÓRIA : ESCREVER | se S=0: RE ← CI<br>se S=1: RE, CI ← CI+1 | | FASE1 ← 1 |

| | INSTRUÇÕES | T0 | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|---|---|
| FASE 4 | TRE | MEMÓRIA : LER SOMENTE<br>1/Lwa | RD ← AC | | MEMORIA: ESCREVER<br>AC ← RE | | RE ← CI | | FASE1 ← 1 |
| | TRI | MEMÓRIA: LER SOMENTE<br>0/Lwa | RD ← AC | | MEMÓRIA : ESCREVER<br>AC ← RE | | RE ← CI | | FASE1 ← 1 |
| | PUG | MEMÓRIA: LER SOMENTE | RD ← CI (4-11) | | MEMORIA: ESCREVER | | RE, CI ← RE+1 | | FASE1 ← 1 |

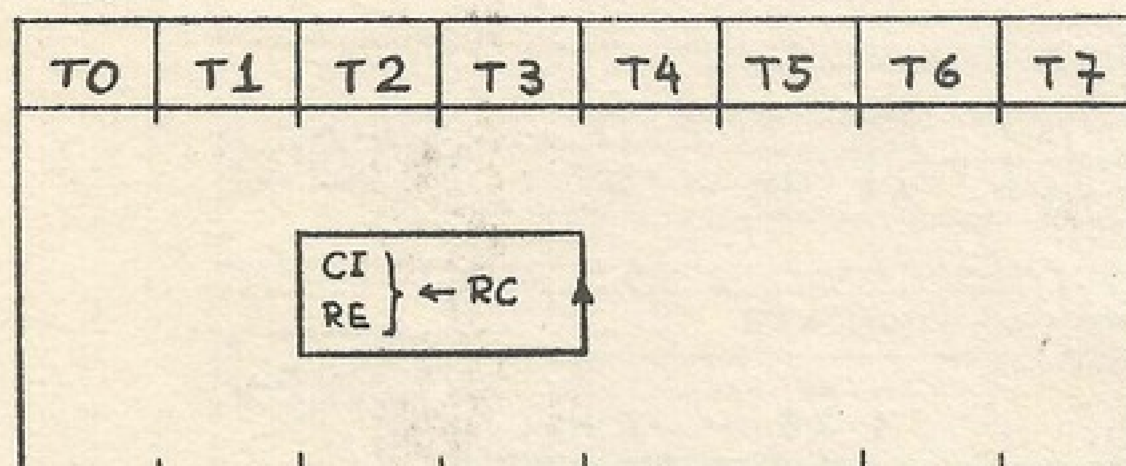
| | INSTRUÇÕES | T0 | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|---|---|
| FASE 5 | CARX<br>ARMX<br>SOMX<br>PLAX | MEMÓRIA : LER E ESCREVER<br>0/Lwa | RD ← Lwbo | | | | RE ← RE + RD<br>se PLI: CI ← RE + RD | | se PLI: FASE1 ← 1<br>SE NÃO PLI: FASE3 ← 1 |

FIGURA 5.15: CARTA DE MICRO-OPERAÇÕES DAS FASES 3, 4 e 5.



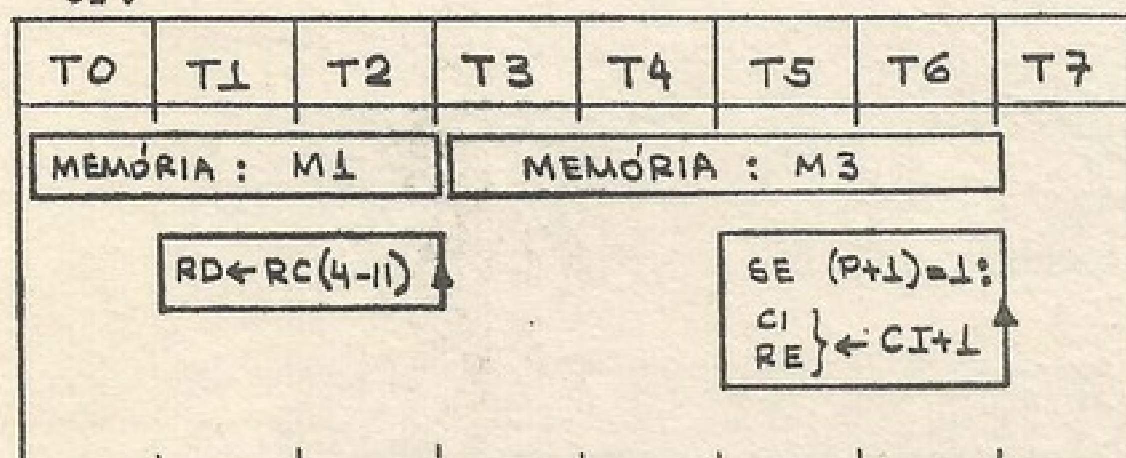

a) sinal PARTIDA

CE:



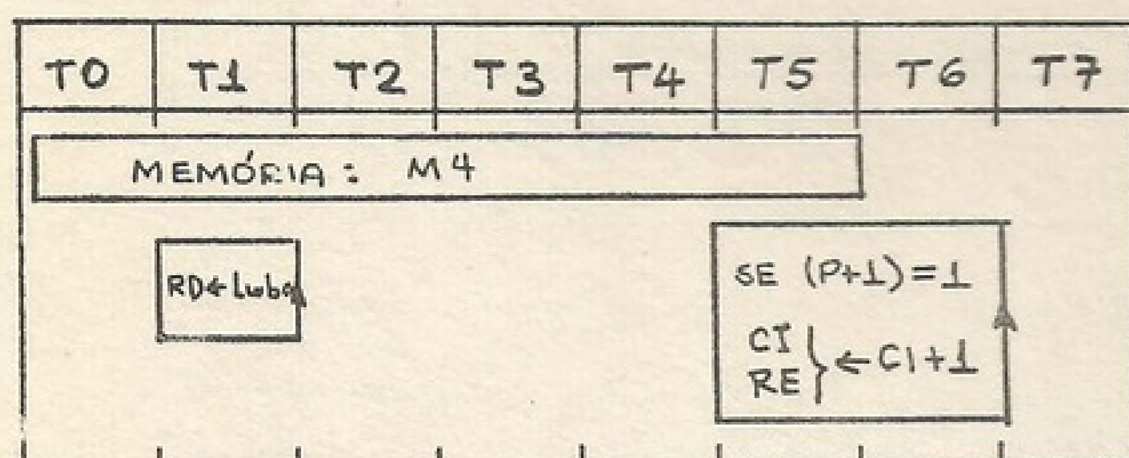

b) Distribuição das microoperações no modo carrega endereço (CE)

CP:




c) Distribuição das microoperações no modo carrega posição (CP)

MP:




d) Distribuição das microoperações no modo mostra posição (MP)

Figura 5.15: MODOS ESPECIAIS DE FUNCIONAMENTO

Nessas cartas de microoperações não se encontram os modos especiais de funcionamento: carrega endereço (CE), carrega posição (CP) e mostra posição (MP). O funcionamento da máquina, nesta situação, ocorre sem o sinal de fase. Utiliza-se a informação de -RODA = 1 para se saber que está parada e o sinal de partida, que dura um ciclo de máquina, substitui o sinal de fase.

Veja na figura 5.15 as cartas de microoperações para este caso. Ali se encontra um sinal chamado (P+1)ou (PM1) que é enviado pelo painel e serve para indicar se é ou não para incrementar o conteúdo do CI.

## 5.7 ÁRVORE DOS SINAIS DE CONTROLE E EXPRESSÕES BOOLEANAS

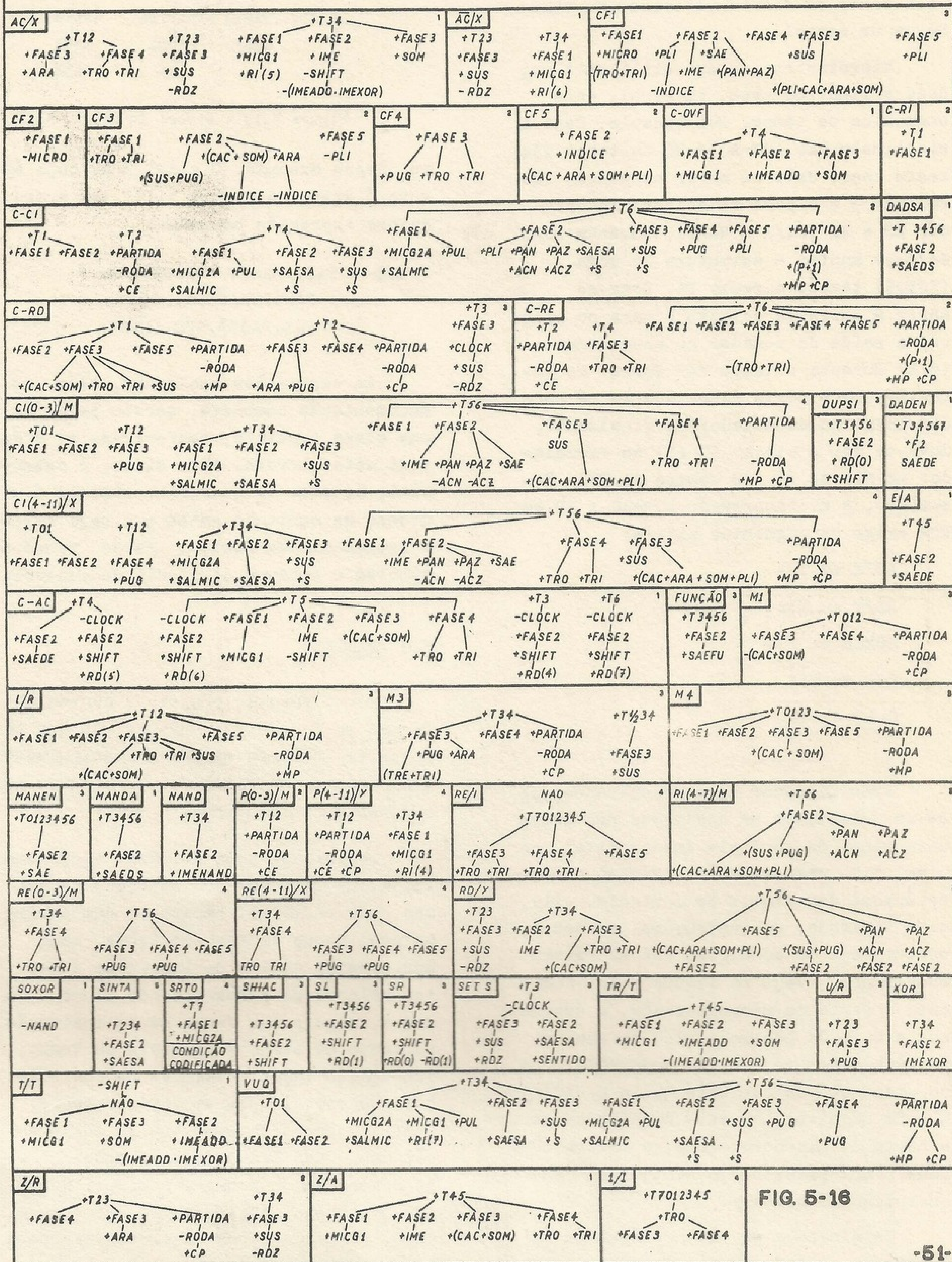
Com a carta de microoperações pronta deve-se realizar a tarefa que muitos autores (19,44) chamam de direto ("straight-forward"): a síntese do circuito. Para isso precisava-se definir os circuitos. Nesta fase, o projeto seguiu as seguintes etapas:

1) Definição dos sinais elétricos que compõe cada microinstrução com seus tempos relativos bem determinados.

2) Relação de todos esses sinais com as condições de sua existência num, gráfico chamado de árvore dos sinais de controle.

3) Determinação da expressão booleana de cada sinal de controle.

A primeira parte, a da definição dos sinais elétricos que sintetizarão cada microinstrução, é uma tarefa que exige do projetista o conhecimento detalhado do fluxo de dados. E, como os projetos correm em paralelo, ele tem a oportunidade de influir no projeto do fluxo de dados.



ÁRVORE DOS SINAIS DE CONTROLE
AC/X 1
+T12 +FASE3 +ARA +FASE4 +TRO +TRI
+T34 +T23 +FASE3 +SUS -RDZ
+FASE1 +MICG1 +RI(5) +FASE2 +IME -SHIFT -(IMEADD·IMEXOR) +FASE3 +SOM
AC/X 1
+T23 +FASE3 +SUS -RDZ
+T34 +FASE1 +MICG1 +RI(6)
CF1 3
+FASE1 +MICRO (TRO+TRI)
+FASE2 +PLI +SAE +IME -INDICE +(PAN+PAZ)
+FASE4 +FASE3 +SUS +(PLI+CAC+ARA+SOM)
+FASE5 +PLI
CF2 1
+FASE1 -MICRO
CF3 2
+FASE1 +TRO +TRI
+FASE2 +(SUS+PUG) +(CAC+SOM) +ARA -INDICE -INDICE
+FASE5 -PLI
CF4 2
+FASE3 +PUG +TRO +TRI
CF5 2
+FASE2 +INDICE +(CAC+ARA+SOM+PLI)
C-OVF 1
+T4 +FASE1 +MICG1 +FASE2 +IMEADD +FASE3 +SOM
C-RI 2
+T1 +FASE1
C-CI 2
+T1 +FASE1 +FASE2
+T2 +PARTIDA -RODA +CE
+T4 +FASE1 +MICG2A +SALMIC +PUL +FASE2 +SAESA +S +FASE3 +SUS +S
+T6 +FASE1 +MICG2A +SALMIC +PUL
+FASE2 +PLI +PAN +ACN +PAZ +ACZ +SAESA +S
+FASE3 +SUS +S +FASE4 +PUG +FASE5 +PLI
+PARTIDA -RODA +(P+1) +MP +CP
DADSA 1
+T3456 +FASE2 +SAEDS
C-RD 3
+T1 +FASE2 +FASE3 +(CAC+SOM) +TRO +TRI +SUS +FASE5 +PARTIDA -RODA +MP
+T2 +FASE3 +ARA +PUG +FASE4 +PARTIDA -RODA +CP
+T3 +FASE3 +CLOCK +SUS -RDZ
C-RE 2
+T2 +PARTIDA -RODA +CE
+T4 +FASE3 +TRO +TRI
+T6 +FASE1 +FASE2 +FASE3 +FASE4 +FASE5 -(TRO+TRI) +PARTIDA -RODA +(P+1) +MP +CP
CI(0-3)/M 4
+T01 +FASE1 +FASE2
+T12 +FASE3 +PUG
+T34 +FASE1 +MICG2A +SALMIC +FASE2 +SAESA +FASE3 +SUS +S
+T56 +FASE1 +FASE2 +IME +PAN +PAZ +SAE -ACN -ACZ
+FASE3 SUS +(CAC+ARA+SOM+PLI)
+FASE4 +TRO +TRI +PARTIDA -RODA +MP +CP
DUPSI 3
+T3456 +FASE2 +RD(0) +SHIFT
DADEN 1
+T34567 +F2 SAEDE
CI(4-11)/X 4
+T01 +FASE1 +FASE2
+T12 +FASE4 +PUG
+T34 +FASE1 +MICG2A +SALMIC +FASE2 +SAESA +FASE3 +SUS +S
+T56 +FASE1 +FASE2 +IME +PAN +PAZ +SAE -ACN -ACZ
+FASE4 +TRO +TRI +FASE3 +SUS +(CAC+ARA+SOM+PLI) +PARTIDA -RODA +MP +CP
E/A 1
+T45 +FASE2 +SAEDE
C-AC 1
+T4 +FASE2 +SAEDE -CLOCK +FASE2 +SHIFT +RD(5)
+T5 -CLOCK +FASE2 +SHIFT +RD(6) +FASE1 +MICG1 +FASE2 IME -SHIFT +FASE3 +(CAC+SOM) +FASE4 +TRO +TRI
+T3 -CLOCK +FASE2 +SHIFT +RD(4)
+T6 -CLOCK +FASE2 +SHIFT +RD(7)
FUNÇÃO 3
+T3456 +FASE2 +SAEFU
M1 3
+T012 +FASE3 -(CAC+SOM) +FASE4 +PARTIDA -RODA +CP
I/R 3
+T12 +FASE1 +FASE2 +FASE3 +(CAC+SOM) +TRO +TRI +SUS +FASE5 +PARTIDA -RODA +MP
M3 3
+T34 +FASE3 (TRE+TRI) +PUG +ARA +FASE4 +PARTIDA -RODA +CP
+T1234 +FASE3 +SUS
M4 3
+T0123 +FASE1 +FASE2 +FASE3 +(CAC+SOM) +FASE5 +PARTIDA -RODA +MP
MANEN 3
+T0123456 +FASE2 +SAE
MANDA 1
+T3456 +FASE2 +SAEDS
NAND 1
+T34 +FASE2 +IMENAND
P(0-3)/M 2
+T12 +PARTIDA -RODA +CE
P(4-11)/Y 4
+T12 +PARTIDA -RODA +CE +CP
+T34 +FASE1 +MICG1 +RI(4)
RE/I 4
NAO +T7012345 +FASE3 +TRO +TRI +FASE4 +TRO +TRI +FASE5
RI(4-7)/M 3
+T56 +FASE2 +(CAC+ARA+SOM+PLI) +(SUS+PUG) +PAN +ACN +PAZ +ACZ
RE(0-3)/M 4
+T34 +FASE4 +TRO +TRI
+T56 +FASE3 +PUG +FASE4 +PUG +FASE5
RE(4-11)/X 4
+T34 +FASE4 TRO TRI
+T56 +FASE3 +PUG +FASE4 +PUG +FASE5
RD/Y 2
+T23 +FASE3 +SUS -RDZ
+T34 +FASE2 IME +FASE3 +(CAC+SOM) +TRO +TRI
+T56 +(CAC+ARA+SOM+PLI) +FASE2 +FASE5 +(SUS+PUG) +FASE2 +PAN +ACN +FASE2 +PAZ +ACZ +FASE2
SOXOR 1
-NAND
SINTA 8
+T234 +FASE2 +SAESA
SRTO 4
+T7 +FASE1 +MICG2A CONDIÇÃO CODIFICADA
SHIAC 3
+T3456 +FASE2 +SHIFT
SL 3
+T3456 +FASE2 +SHIFT +RD(1)
SR 3
+T3456 +FASE2 +SHIFT +RD(0) -RD(1)
SET S 3
+T3 -CLOCK +FASE3 +SUS +RDZ +FASE2 +SAESA +SENTIDO
TR/T 1
+T45 +FASE1 +MICG1 +FASE2 +IMEADD -(IMEADD·IMEXOR) +FASE3 +SOM
U/R 2
+T23 +FASE3 +PUG
XOR 1
+T34 +FASE2 IMEXOR
T/T 1
-SHIFT NAO +FASE1 +MICG1 +FASE3 +SOM +FASE2 +IMEADD -(IMEADD·IMEXOR)
VUQ 3
+T01 +FASE1 +FASE2
+T34 +FASE1 +MICG2A +SALMIC +MICG1 +RI(7) +PUL +FASE2 +SAESA +FASE3 +SUS +S
+T56 +FASE1 +MICG2A +SALMIC +PUL +FASE2 +SAESA +S +FASE3 +SUS +S +PUG +FASE4 +PUG +PARTIDA -RODA +MP +CP
Z/R 8
+T23 +FASE4 +FASE3 +ARA +PARTIDA -RODA +CP
+T34 +FASE3 +SUS -RDZ
Z/A 1
+T45 +FASE1 +MICG1 +FASE2 +IME +FASE3 +(CAC+SOM) +FASE4 +TRO +TRI
1/1 4
+T7012345 +TRO +FASE3 +FASE4

FIG. 5-16

Aqui, o projetista tem que analisar microinstrução por microinstrução, e relacionar os sinais elétricos no tempo. Segue um exemplo:

Microinstruções que utilizam a unidade aritmética usam, em geral, dois segmentos de tempo. Por exemplo: Seja a microinstrução AC←AC + RD no tempo T34. Neste caso, deve-se abrir as portas de seleção 3 e 2 para se colocar no somador AC e RD (AC/X e RD/Y), durante T34; deve-se manter a estrutura de soma (SOXOR) também durante T4. Deve-se abrir a porta de seleção 6 para se colocar a saída do somador no acumulador (Z/A) durante o tempo T45 (para se garantir que a porta fique aberta na hora do "clock" no acumulador) e finalmente, deve-se dar o "clock" (C-AC) no acumulador no fim de T4 (ou começo de T5). Resumindo, a microoperação AC←AC + RD em T34 exige os seguintes sinais:

AC/X em T34

RD/Y em T34

SOXOR em T34

Z/A em T45

C-AC em T5

Após dissecar as microoperações, pode-se relacionar as condições nas quais cada sinal de controle deverá estar no nível "1". Essa relação recebeu o nome de árvore dos sinais de controle. Ali, está indicado, as instruções, as fases, os tempos e as condições que energizam cada sinal. Veja na figura 5.16. Assim como a carta de microoperação, a árvore dos sinais de controle além da sua importância no projeto, é fundamental na documentação do que está feito e definido. Em um nível bem mais detalhado que a carta de microoperações, a árvore define o comportamento do sistema em cada instrução e instante.

Terminada a árvore, está-se em condições de se escrever as funções booleanas de cada sinal de controle. É imedia-

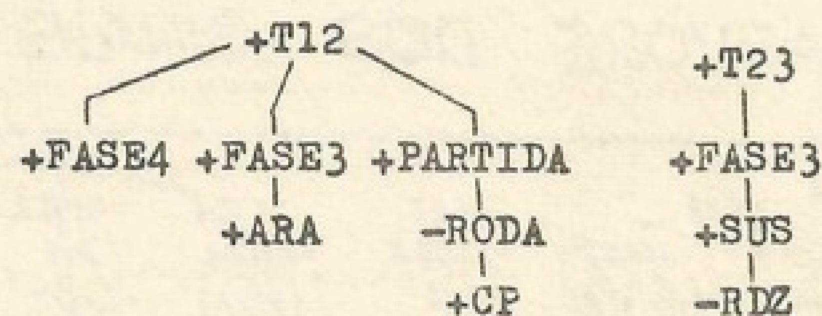



Figura 5.17 : árvore do sinal +Z/R

to. Segue exemplo. O sinal Z/R, cuja árvore é vista na figura 5.17, tem a seguinte expressão booleana:

$$Z/R = T12.(FASE4 + FASE3.ARA + PARTIDA.(\overline{RODA}).CP) + T23.FASE3.SUS.(\overline{RDZ})$$

As expressões booleanas, além da documentação compacta, servem para, em uma etapa seguinte, conferir-se se a árvore está correta. Além disso, é essencial, no caso de querer-se simular a unidade de controle antes que seja feita a implementação física. Essas funções simulam o gerador de sinais de controle.

## 5.8 TESTES

Nesta fase do projeto é conveniente que se faça alguns testes, já que se passaram inúmeras etapas de manipulação dos dados e, provavelmente, inúmeros erros foram introduzidos.

Decidiu-se aqui utilizar o sistema IBM 1130. De posse das equações booleanas, não é difícil pensar-se num programa que, dadas certas condições, como por exemplo código de instrução e certos valores de parâmetros do fluxo de dados, siga pelas fases dessa instrução segmento de tempo a segmento de tempo, imprimindo todos os sinais de controle que são energizados em cada instante. Depois disso, com uma simples comparação com a carta de microoperações pode-se descobrir os possíveis erros. Na figura 5.18 encontra-se a listagem dos sinais, fornecidos por esse programa para o caso da instrução PUG na fase 1.

PROJETO COMPUTADOR CONTROLE
EQUACOES BOOLEANAS DO TIMING

INSTRUCOES E CONDICOES

```
***************************************
PUG
FASE 1
***************************************
T0
 CI3BM
 CI4BX
 VUQ
 RBI
 TBT
 M4
 SOXOR

T1
 CI3BM
 CI4BX
 VUQ
 LBR
 RBI
 TBT
 CKCI
 CKRI
 M4
 SOXOR

T2
 RBI
 TBT
 M4
 SOXOR
 LBR

T3
 RBI
 TBT
 M4
 SOXOR

T4
 RBI
 TBT
 SOXOR

T5
 CI3BM
 CI4BX
 RBI
 TBT
 SOXOR

T6
 CI3BM
 CI4BX
 RBI
 TBT
 CKRE
 SOXOR

T7
 CF2
 RBI
 TBT
 SOXOR
```

**Figura 5.18:** Amostra da listagem do programa de teste.

Da experiência obtida, pode-se dizer que o programa seria mais eficiente se imprimisse as microoperações ao invés dos sinais de controle.

Essa foi uma solução adotada para se contornar a necessidade de simulação da unidade de controle que seria a solução ideal para este estágio do projeto.

## 5.9 SÍNTESE DOS CIRCUITOS

A partir da árvore dos sinais de controle deve-se sintetizar os circuitos respeitando-se os seguintes vínculos:

- Família 7400 de circuitos integrados TTL.

- atraso máximo em relação aos sinais de tempo.

- "fan-out" máximo.

Esse circuito, com 52 saídas e 93 entradas deve ser sintetizado tendo por ótimo o circuito de menor quantidade de blocos lógicos ("gates"). Como o circuito não cabe em apenas uma placa de circuito impresso, o seu projeto inicial, como um só circuito provocará sérios problemas de partição, já que a quantidade de pinos no conector de entrada e saída é limitada.

Fez-se um primeiro anteprojeto da síntese num bloco só. Assim, pode-se agrupar os sinais de controle em cuja síntese existia forte interação. A etapa seguinte foi a partição (discutida no item seguinte) na qual se definiu quais sinais seriam gerados na mesma placa de circuito impresso. Então estava-se pronto para o projeto detalhado e final do circuito gerador de sinais de controle.

Em média, em cada placa, foram colocados 12 sinais de controle, com 45 entradas. Um circuito desse tamanho torna ineficiente as técnicas de síntese.

Nùm processo de tentativa e correção, a síntese do circuito evoluiu para a sua forma final, que evidentemente não é ótima.

Problemas particulares foram resolvidos. Por exemplo, os sinais C-CI e C-RE (clock nos registradores CI e RE ) devem ter alto "fan-out", o que poderia ser resolvido colocando-se um circuito "buffer" na saída. Mas como os sinais de "clock" nos registradores são gerados com apenas 1 nível de atraso isso se tornou impossível. O que se fez foi gerar esses sinais em dois pontos, paralelamente, e chamá-los de C-CI1 e C-CI2, e C-RE1 e C-RE2. Sempre que inevitável utilizou-se o sinal chamado "CLOCK" que vem do relógio central para provocar-se alguma ação no meio de um segmento de tempo (veja por exemplo o sinal C-AC no caso de deslocamentos).

As várias condições de energização dos sinais em tempos diferentes foram geradas separadamente para, no final juntá-las num bloco "AND-NOR" implementado com a técnica "DOT-OR"[10]. Com isso, pode-se economizar um nível lógico, importante para os sinais de "clock" nos registradores. Na figura 5.19 existe um circuito típico gerando o sinal AC/X.

## 5.10 PARTIÇÃO

### 5.10.1 ASPECTOS GERAIS

Giannuzzi[48] define partição ("partitioning") como sendo o processo de subdividir um bloco representando o circuito lógico de um sistema em subblocos cada um representando a lógica que pode ser implementada em um dado elemento de empacotamento.

Apesar desse problema existir em vários níveis, tratar-se-á aqui apenas do problema da partição de um circuito em pedaços que seriam alojados em placas de circuito impresso. Nesse processo existem três vínculos básicos: número máximo de componentes por placa, máxima potência dissipada por placa e número máximo de sinais de entrada e saída na placa.

Número máximo de componentes por placa

É um problema físico. Cada placa tem suas dimensões fixas e padronizadas, então existe um número máximo de componentes possíveis de serem alojados na placa. A partição feita deve satisfazer a essa restrição.

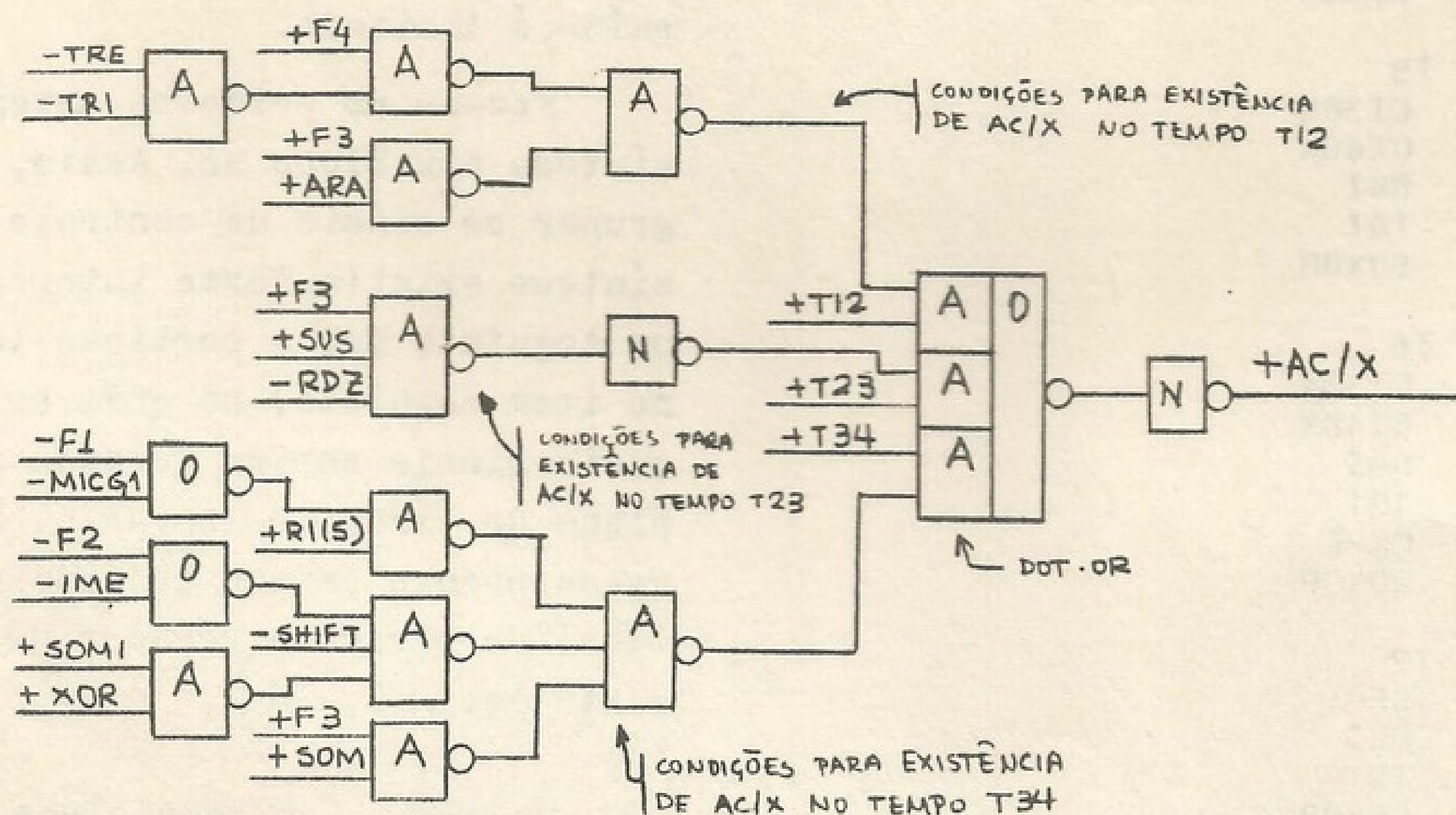



Figura 5.19: Exemplo de um sinal do gerador de sinais de controle

Potência máxima dissipada

Essa grandeza está associada ao processo de refrigeração ("cooling") do circuito e à corrente de alimentação máxima fornecida pelo conector de entrada e saída. Na partição, a potência dissipada pelo circuito especificado para cada placa, deve, é claro, dissipar menos que esse máximo.

Número de pinos de entrada e saída máximo

É outro problema físico. Padronizando-se o conector de entrada e saída da placa, existe um número máximo (igual ao número de pinos do conector) de sinais de entrada/saída possíveis de serem colocados em uma placa.

Até então não se tem notícias de um algoritmo de partição ótimo, que possa ser mecanizado (executado por um computador). O que se tem feito são processos quase ótimos.

5.10.2 ALGORITMO DA PARTIÇÃO DOS CIRCUITOS DO GERADOR DOS SINAIS DE CONTROLE

O problema particular da partição dos circuitos do "gerador dos sinais de controle" foi resolvido com um algoritmo que exige a interação homem/máquina. Como, dada as características deste projeto, era mais importante ter-se, rapidamente, uma solução boa do que ter-se uma solução ótima muito tempo depois, o problema ainda continua em aberto já que o algoritmo adotado está longe de ser considerado bom.

Do anteprojeto do circuito descobriu-se que dos três vínculos já citados, o mais forte era o do número máximo de sinais de entrada/saída. Isto é , quando se garantir que o número dos sinais de entrada/saída não ultrapassam o limite, estará automaticamente garantido que os outros vínculos serão satisfeitos. Utilizou-se conectores com 70 pinos e reservou-se 15% deles para futuras modificações, assim, fez-se a partição limitando-se em 60 o número de sinais de entrada/saída.

Os circuitos foram divididos em blocos que não seriam separados pela partição. Cada bloco gera um sinal de controle. O problema então, passou a ser o da distribuição desses blocos em um certo números de placas, de modo que esse número seja mínimo.

O algoritmo exige que se defina, "a priori" um certo número de placas de circuito impresso, e que em cada uma delas coloque-se alguns blocos (sementes) para se iniciar o processo. Fornece-se as descrições (entradas e saídas) dos outros blocos que devem ser distribuídos nas placas.

O algoritmo caminha, varrendo todas as placas e incluindo os novos blocos. Em primeiro lugar, ele inclui os blocos cujas entradas são as mesmas que as do circuito que já está na placa. Esgotados os blocos com essa característica, o algoritmo admite uma entrada diferente e depois duas, três e assim por diante.

O trabalho mais importante no processo é o do projetista que deve, no anteprojeto dos circuitos, definir conjuntos de blocos que seriam as sementes. Cada conjunto deve ser composto de sinais que em cuja geração podem compartilhar grande parte dos circuitos, e dois conjuntos devem ser caracteristicamente diferentes. O projetista começa a partição tentanto um número pequeno de placas, e vai aumentando até que consiga um resultado satisfatório.

É lógico que esse algoritmo pode ser melhorado e semi automatizado. Pensou-se inclusive em gerar, automaticamente, as sementes. O operador forneceria apenas o número de placas. Mas, a opinião do autor é que não se deve melhorar este algoritmo e sim desenvolver outro, mais geral, que sirva para qualquer tipo de circuitos, levando-se em conta outros vínculos.

5.10.3 RESULTADOS OBTIDOS

Obteve-se um bom resultado com a partição em 5 placas, relacionadas a seguir.

| | |
|---|---|
| CTI-4 | prancha XII |
| CT2-5 | prancha XIII |
| CT3-6 | prancha XIV |
| CT4-7 | prancha XV |
| CT5-8 | prancha XVI |

Essas placas, foram montadas com a técnica de "wire-wrap" (fio enrolado) , conveniente na implementação de protótipos, onde se necessita de facilidades de mudanças.

5.11 DOCUMENTAÇÃO

"Dada a complexidade do sistema e do número de pessoas nele envolvidas, todas as fases do projeto foram documentadas, desde as reuniões iniciais sôbre a definição da arquitetura, até as fases finais como os "layouts" dos circuitos impressos, foram registrados em cadernos especiais, listagens de computador, gráficos, etc".[49]

Além da documentação de projeto, como carta de microoperações, árvore dos sinais de controle, etc, cada placa de circuito é acompanhada pela seguinte documentação básica:

1) Diagrama em blocos da lógica: é um desenho esquemático, como o da figura 4.12, ilustrando a lógica.

2) Diagrama detalhado em termos dos CI's: são as pranchas apresentadas. São desenhos que trazem, além das informações da lógica, informações sôbre as ligações realizadas em cada placa, posição dos integrados, sinais nos pinos dos conectores, etc.

3) Posição dos CI's na placa: é um esquema bastante simples, indicando apenas, nas posições padronizadas quais os CI's que deverão ser colocados. É um documento preparado pelo projetista, para a equipe de montagem.

4) "Layout" do circuito impresso: um esquema mostrando as interligações dos pontos nas placas de circuito impresso. A partir dele é preparada a arte final que será fotografada e reproduzida na placa.

5) Listagem de "wire-wrap": acompanha as placas montadas com essa técnica. São preparadas por um programa rodado no sistema 1130. São usadas pela equipe de montagem para fazer as interligações na placa.

6) Lista dos sinais nos conectores de entrada e saída da placa: é uma simples relação, usada mais tarde na preparação da lista de interligação das placas.

Essa documentação se mostrou eficiente sendo os mais usados os diagramas detalhados da lógica.

A experiência mostrou que a nomenclatura adotada tinha algumas deficiências. Os nomes dos sinais elétricos foram escolhidos seguindo a seguinte regra básica:

1) Nome de sinais que abrem portas têm uma barra (/) com o nome de entrada e de saída. Exemplo AC/X, Z/R.

2) Os sinais de "clock" em registradores são escritos com o prefixo "C-" na frente do nome do registrador. Exemplo C-CI, C-RI.

3) Outros nomes devem ser mnemônicos, ou seja devem lembrar a função do sinal.

4) Os sinais são acompanhados por sinais + ou - , conforme sua polaridade seja direta ou invertida.

5) Os bits dos registradores são escritos com o nome do registrador seguido pelo número do bit entre parêntesis. Exemplo: AC(3), RE(11). Conjuntos de bits são escritos colocando-se entre parêntesis os números do primeiro e último bits do conjunto. Exemplo AC(0-3) CI(4-11).

Falando-se em termos de unidade de controle, esta estrutura de nomes necessita de um número, indicando onde (em que placa, em que módulo) o sinal foi gerado.

A documentação também falha na indicação do "fan-out" já utilizado de cada sinal. É uma informação útil para a avaliação do atraso e qualidade desse sinal, sendo também necessária no caso de usar-se esse sinal em alguma modificação.

Os maiores problemas encontrados na documentação foram provocados pelas mudanças dos nomes. Por exemplo, no início do projeto adotou-se para os flip flops de vai-um e transbordamento os nomes TRANSBORDAMENTO (T) e "OVERFLOW" (OVF), respectivamente. Posteriormente esses nomes foram alterados para os nomes atuais. Isso dificulta muito a análise dos circuitos por uma pessoa que desconheça os detalhes do projeto.

5.12 SUMÁRIO

Do projeto da unidade de controle concluiu-se que a automatização do projeto, apesar de ser a meta da ciência do projeto lógico, está longe de ser atingida. A presença do projetista é decisiva. No caso analisado, praticamente não se automatizou nada, apenas alguns testes. Ficou então perfeitamente caracterizado o aspecto heurístico do projeto. O relógio central e o controle de estado foram projetados procurando-se resolver separadamente cada problema. A escolha de cada detalhe dependeu apenas da experiência do projetista. O circuito gerador de sinais de controle, dada a sua complexidade exigiu o desenvolvimento de uma técnica que ajudasse o projetista circundar todas as variáveis. Existiram as seguintes etapas: determinação da carta de microoperações, definição da árvore dos sinais de controle, anteprojeto do circuito, partição em placas e projeto detalhado da lógica.

A experiência mostrou ainda que a estrutura da documentação do projeto exige algumas modificações.

# 6- OBSERVAÇÕES FINAIS

## 6.1 DEPURAÇÃO DO SISTEMA

Como os principais comentários sobre a técnica adotada no projeto foram feitos durante a descrição deles, para este capítulo ficou reservado a análise dos principais erros encontrados na fase de depuração ("debug") do sistema.

A depuração foi feita com a seguinte técnica:

Montou-se as placas e testou-se cada uma delas individualmente em um painel de testes. Esse teste individual poderia ter sido feito no computador HP 2116 que oferece recursos para esse tipo de trabalho. Porém, o tempo gasto na preparação dos programas de testes seria excessivo e já que se tinha que testar apenas uma ou duas placas de cada tipo, a melhor solução foi o teste manual em um painel. Nesse painel varia-se as entradas da placa e as saídas são mostradas em lâmpadas pilotos. Usou se um voltímetro para se verificar pontos internos à placa. O teste foi feito tendo-se em mãos, como referência, o diagrama detalhado da lógica (pranchas).

Após testar-se todas as placas e corrigir-se todos os erros encontrados, pode-se começar o teste do conjunto, interligando-se as placas. Se se ligasse todas as placas de uma só vez, ter-se-ia uma quantidade muito grande de erros para corrigir, o que dificultaria o trabalho. Por isso iniciou-se com algumas placas e foi-se aumentando o sistema à medida que os erros iam sendo corrigidos.

Iniciou-se os testes pelos cartões do painel. Interligou-se as chaves e pilotos do painel aos cartões e fez-se o conjunto funcionar. Após isso colocou-se a placa de relógio central e mediu-se os sinais no painel trazeiro para a confirmação de que eles estavam corretamente distribuídos. Colocou-se então o cartão de contole de estado. Nesse ponto pode-se testar todos os modos de funcionamento forçando-se pelo painel trazeiro os bits dos registradores RD e RI, testar-se uma parte importante do controle: a sequencialização das fases.

Neste ponto não se teve escolha:colocou-se os cartões do fluxo de dados de uma só vez - com exceção da memória que foi o último componente a ser inserido na UCP.

Com a unidade de controle e o fluxo de dados (sem a memória) interligados, pode-se testar grande parte do sistema, bastou utilizar-se de um conjunto de 8 chaves que simulava a saída da memória. Nesta fase foi possível o teste detalhado da carta de microoperações, pois com o registrador de 8 chaves citado, pode-se simular todas as instruções e, de ciclo a ciclo, verificar o seu funcionamento.

Pronta essa etapa, restou apenas colocar-se a memória e testar-se a unidade central completa, em toda a sua potência, com sinais em todas as linhas.

Após isso, o sistema foi entregue à equipe de "software"que, ao programar o sistema, indicou alguns erros ainda existentes no sistema, principalmente os de caráter intermitente, que foram os mais comuns nesta etapa da depuração.

Gastou-se, aproximadamente, 300 homen horas para se depurar a unidade central de processamento, a partir das placas já testadas individualmente.Esse tempo cresceu além do esperado porque, na etapa seguinte, ao se inserir os cartões de interface no sistema, foram necessárias algumas mudanças na UCP.

## 6.2 ANÁLISE DOS ERROS

Segue uma análise dos erros mais comuns:

PREPARAÇÃO DE DADOS: a maioria dos erros encontrados foram falhas na preparação da documentação. Por exemplo, a quase totalidade das ligações erradas foram provocadas por erros na listagem de interligações que estava errada por causa dos dados fornecidos ao programa. Uma maneira de se diminuir essas falhas seria colocar-se em todos os programas rotinas de testes dos dados. Esses erros de documentação foram provocados, principalmente, pela própria deficiência dela. Quanto mais redundante for a documentação, menor será a possibilidade de propagação de erros, além dos erros poderem se facilmente detetados.

LÓGICA: estas falhas existiram em menor freqüência, provocadas principalmente por deficiências de documentação. Esses erros não existiriam se os circuitos, depois de projetados fossem simulados. A simulação poder-se-ia dizer, foi a ferramenta de projeto lógico que mais fez falta durante este projeto.

DEFINIÇÃO DA ARQUITETURA: São as falhas mais árduas de se corrigir. Elas podem exigir grandes mudanças. Dentre os problemas ocorridos, pode-se ressaltar que a impossibilidade de limpar a interrupção pelo painel, obrigou a se colocar a chave de preparação ("preset") que liga o sinal LAL. Muitos erros, neste nível, também poderiam ser corrigidos antes da montagem, caso se dispusesse de um simulador no nível de sistema.

"HAZARDS": apesar de existirem com menor frequência que a esperada, eles não deixaram de ocorrer. Todos os erros desse tipo deveriam ser corrigidos durante a fase de projeto. O projetista deve evitar colocar num mesmo bloco lógico dois sinais de tempo (foram os mais comuns), sem se assegurar de que não existirá "hazards". É uma falha inerente aos atrasos dos blocos lógicos e seria necessário um programa de simulação bastante requintado para que os detete.

ALTERAÇÕES: uma falha comum foi provocada pelas próprias correções. Ao se alterar os circuitos para se corrigir alguma falha, essa alteração poderia provocar falhas em outra parte. Uma documentação eficiente e o completo conhecimento das partes envolvidas na correção são fundamentais para ajudar o projetista evitar esses erros.

OUTROS TIPOS DE FALHAS: outros tipos de falhas foram de ordem tecnológica ou de implementação física. Por exemplo, um mau funcionamento de um flip flop era provocado pela sua entrada de SET não usada, que fora deixada em aberto. Após ligar-se esse ponto na fonte de +5V o problema deixou de existir. Outro exemplo: a saída de um bloco lógico estava presa no zero por causa de uma interrupção no circuito impresso que impedia a alimentação daquele integrado.

## 6.3 - CONCLUSÕES

Pretendeu-se neste trabalho descrever o projeto lógico da unidade de controle do minicomputador do LSD. O que aqui se disse não ficou apenas no papel, O sistema foi montado e testado. A técnica de projeto também foi testada. As conclusões tiradas e observações mais importantes foram enunciadas no capítulo 5.

Apesar de todo o trabalho ter sido desenvolvido e posto em prática utilizando-se circuitos integrados da família SSI ("small scale integration")

acredita-se que com pequenas mudanças, em detalhes apenas, isso fica sendo válido em famílias mais complexas. Se por exemplo usar-se os circuitos sugeridos por Lowenschuss[50], onde a idéia é utilizar-se um único que sintetizaria todas as funções lógicas de um certo número de entradas, ter-se-ia variações na técnica apenas no item do projeto lógico dos circuitos (projeto detalhado), talvez mudassem os vínculos da partição. Mas a técnica não seria alterada.

# 7-APÊNDICE

Colocou-se neste apêndice possíveis modificações e expansões do sistema. Esses detalhes são os que parecem ser mais convenientes, sem que seu custo seja proibitivo. São modificações que foram sugeridas depois do sistema posto em funcionamento, sentindo-se a necessidade delas ao se utilizá-lo.

## 7.1 RESTAURAÇÃO AUTOMÁTICA DE V e T

Quando chega um pedido de interrupção, a primeira providência tomada é salvar o estado do sistema, isto é, armazenar na memória os conteúdos dos registradores AC, V e T. Terminado o atendimento da interrupção, o estado é restaurado. Tudo isso é feito por "software". A parte mais longa dessas rotinas é o tratamento dos flip flops V e T, que poderia ser economizada caso se fizesse a salvação e restauração deles por "hardware".

Isso pode ser conseguido com a inclusão de dois flip flops ($V_I$ e $T_I$). No início da interrupção (pode-se usar o sinal FORÇA-PUG) os estados dos flip flops V e T são registrados, respectiva

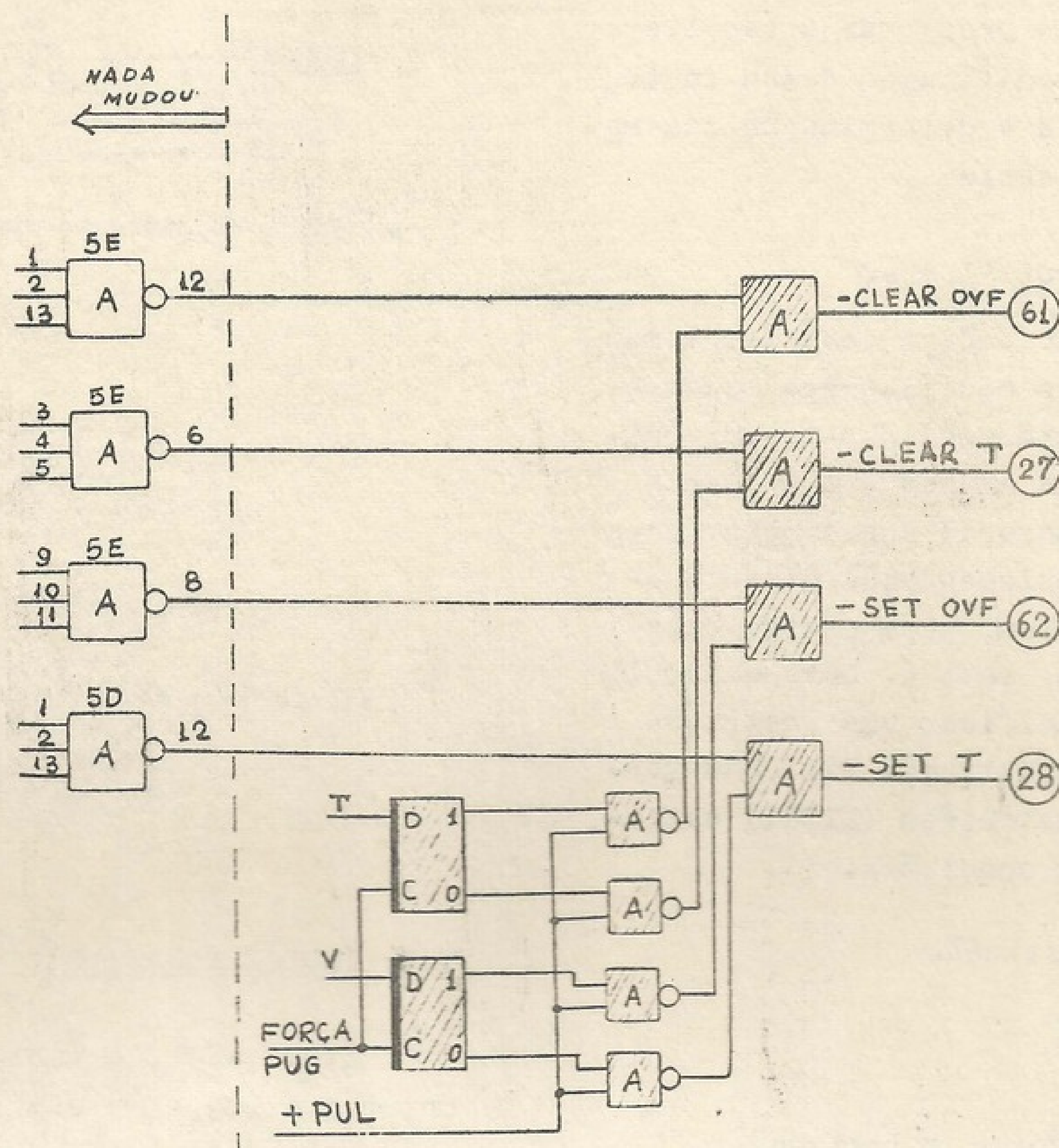



Figura 7.1 : Circuito proposto para o sinal "SRTO" (prancha XV) que provê a restauração automática de V e T.

mente em $V_I$ e $T_I$. No final da interrupção (pode-se usar o sinal PUL) os estados dos flip flops $V_I$ e $T_I$ são devolvidos aos flip flops V e T.

Necessita-se modificar os sinais -SET-T, -CLEAR-T, -SET-OVF e -CLEAR-OVF Na prancha XV vê-se o circuito atual que gera esses sinais, e na figura 7.1 o novo circuito que provê a restauração automática. Essa solução, apesar de não ser a mais barata, parece ser a que exige menos alterações no sistema.

## 7.2 NOVA INSTRUÇÃO: SOMA $V_A$ NO ACUMULADOR

No tratamento de endereços e em aritmética de dupla precisão, essa instrução parece ser essencial. Ela não existe no repertório atual, mas, sem muito custo pode ser implementada.

Existem dois problemas a resolver: o primeiro é a codificação dessa instrução e o segundo é a determinação das modificações necessárias.

### 7.2.1 SÔBRE A CODIFICAÇÃO

A escolha do código está diretamente relacionada às modificações necessárias. É importante escolher-se um código que pertença a grupos e sub grupos que já gerem a maioria dos sinais necessários para a implementação dessa instrução. Outra coisa, esse código deve estar disponível, isto é, deve estar livre ou deve especificar uma instrução de pouco uso. A instrução PNL(2) do grupo 1 das microinstruções (MICG1) satisfaz a essas duas condições.

Então, a instrução

$$\overbrace{\underbrace{1000}_{\text{MICG1}}1010}^{\text{RI}}$$

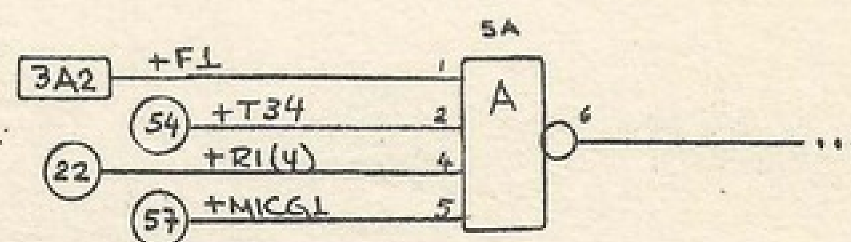
em vez de especificar a instrução [AC] ← [RC(3-11)] + [AC] passa a especificar a nova instrução [AC] ← [AC] + [$V_A$]. Para passar-se daquela instrução para esta basta realizar-se duas coisas:

‖ inibir o sinal P(4-11)/Y
‖ fazer o sinal VUQ igual a $V_A$

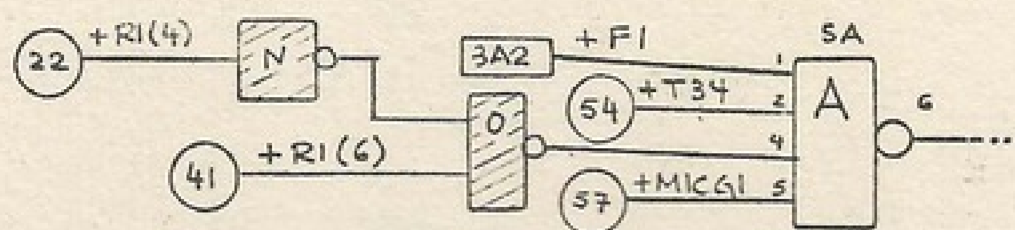
Então, bastaria detetar-se a presença dos dois "uns", em RI(4) e RI(6), para se tomar as duas providências acima. Dessa maneira, além dessa instrução, ter-se-á como produto secundário dessa mudança, a seguinte alteração: a instrução 100001110 que era [AC]←[RC(4-11)] -1 passa a ser [AC]←[$V_A$] -1, que parece ser mais útil que a primeira.

### 7.2.2 SÔBRE AS MODIFICAÇÕES

As modificações ocorrem com os sinais P(4-11)/Y (prancha XV) e VUQ (prancha XVI). A figura 7.2 ilustra a modificação necessária no sinal P(4-11)/Y e na figura 7.3 no sinal VUQ.




a) Parte do circuito atual

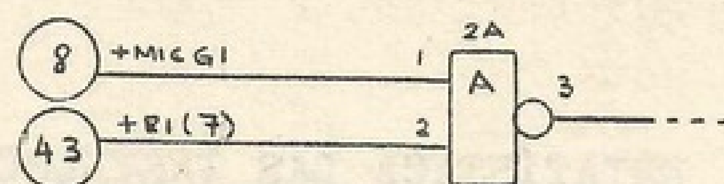



b) Modificações nessa parte

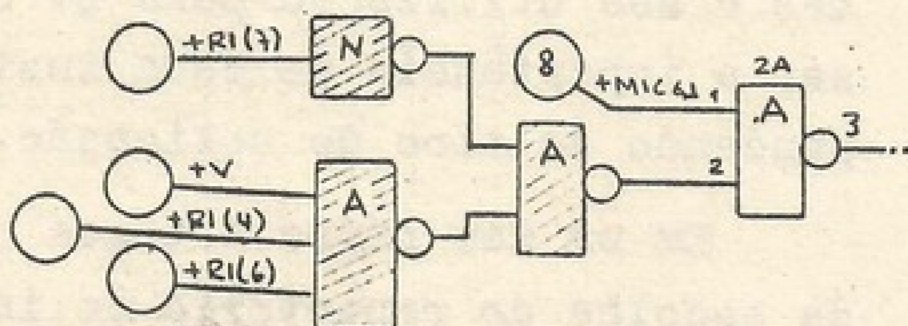
Figura 7.2: Modificações no sinal "P(4-11)/Y (prancha XV) para implementar a instrução AC  AC + $V_A$

## 7.3 ÍNDICE NEGATIVO

Atualmente, o índice (posição de memória zero) é interpretado como um número sem sinal, utilizando-se os 8 bits como amplitude. Parece ajudar um pouco a programação do sistema caso se defina




a) Parte do circuito atual




b) Modificação nessa parte

FIGURA 7.3: Modificações no sinal "VUQ" (prancha XVI) para implementar a instrução AC AC + $V_A$

o bit mais significativo como sinal e assim, em vez de ter-se o índice variando de 0 a 255, ter-se-á a seguinte escala: -128 a +127.

Isso é conseguido facilmente com pequenas alterações na placa FMI-6, onde existem os circuitos do mais 1 (prancha IV) e na placa FS2-4 (prancha II), onde existem os circuitos do somador. A solução é prolongar-se o bit do sinal do índice nos 4 bits seguintes formando um campo de 12 bits. Então, na placa FS2-4, deve-se encaminhar para um pino de saída o sinal +Y(0), gerado no ponto 5C8, e através do painel trazeiro levar se esse sinal para a placa FM1 6, onde ele seria ligado aos quatro bits da entrada do somador (3A4, 3A7, 3A11 e 3A16) que atualmente estão ligados na terra. Essa alteração não causa problemas nas outras situações onde se usa o circuito mais um.

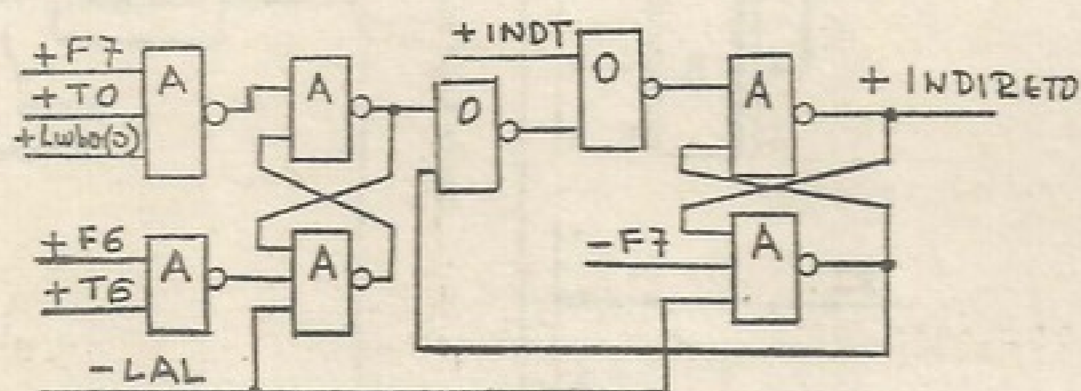



Figura 7.4: Controle do modo indireto

## 7.4 ENDEREÇAMENTO INDIRETO

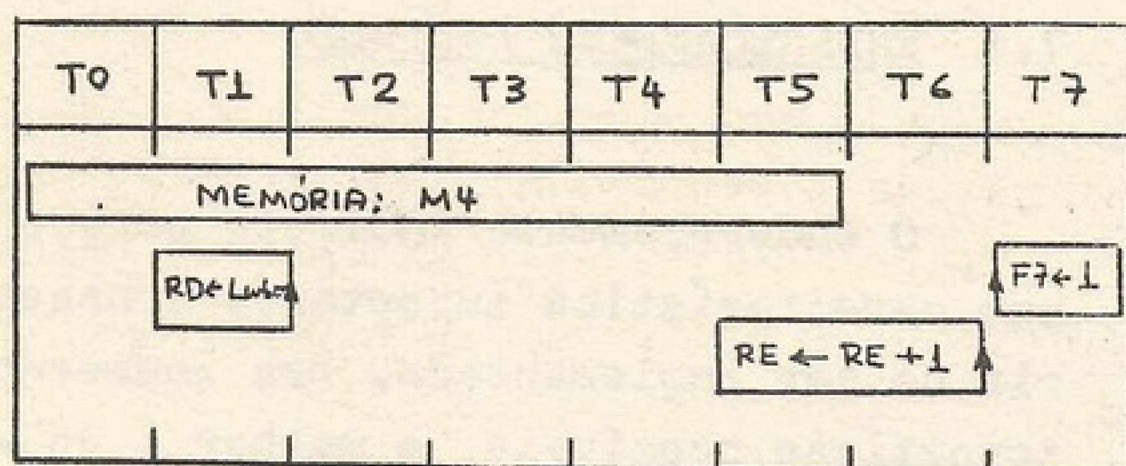
O endereçamento indireto parece ser uma característica importante e necessária de ser implementada. Das inúmeras alternativas possíveis, a melhor é aquela que permite a utilização simultânea do - endereçamento indexado e indireto. Esta é uma grande modificação no sistema e exige extensas alterações, cuja descrição ocuparia, aqui, muito espaço. Por isso , este item será menos detalhado que o anterior.

A melhor alternativa de codificação é a seguinte: com a última instrução das MICG2A(que se chamará "INDT"), especifica-se que a instrução seguinte é de endereço indireto, criando-se assim as instruções "super-longas" - de três palavras. O ponteiro que está gravado na memória deve ocupar doze bits de duas palavras, restando portanto 4 bits para se continuar o endereçamento indireto.

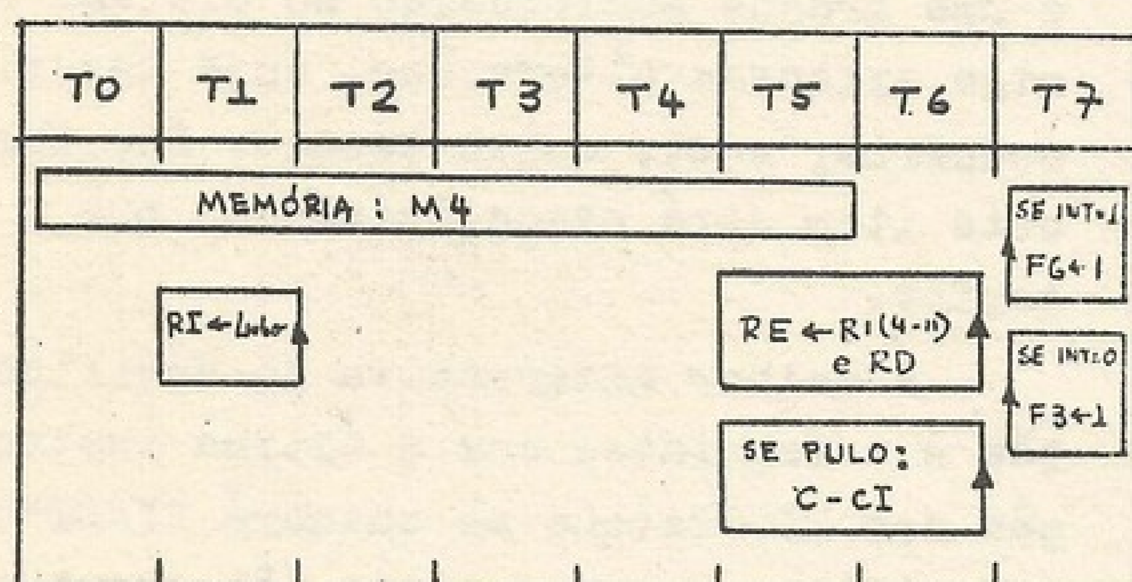
O primeiro problema que surge após ter-se resolvido o da codificação, é o da sequencialização dos ciclos, ou seja, o controle das fases. A solução proposta aqui é a criação de duas novas fases FASE6 e FASE7 (dois novos flip flops na placa de controle do estado) - essas fases seriam usadas para ler-se o endereço na memória (2palavras) e colocar-se no RE. A sequencialização fica resolvida com a redefinição dos sinais CF1 a CF5 e a geração dos dois novos: CF6 e CF7.

Um flip flop (INDIRETO) é ligado com a presença da instrução "INDT" e é limpo com a fase 7. O bit mais significativo do endereço, que especifica se é ou não para continuar o endereçamento indireto, ligará um flip flop que inibirá a limpeza do INDIRETO (figura 7.4).

Providenciada a nova sequencialização das fases, faltaria apenas a modifi




a) FASE 6




b) FASE 7

FIGURA 7.5: Carta de microoperações das fases 6 e 7 (modo indireto)

cação dos sinais de contrôle, com a inclusão dos ciclos de indireto (Fases 6 e 7). A figura 7.5 mostra a carta das microoperações a ser implementada. Para que não se altere o código da instrução RI(0-3) , ao se carregar RI com a palavra mais significativa do endereço, é necessário que se use o sinal FASE7 para se inibir o "clock" nos 4 bits mais significativos do RI.

Mais dois detalhes precisam ser cuidados:

1º - Deve-se impedir que a interrupção ocorra entre a instrução de "INDT" e a seguinte. Para isso, pode-se, na placa de controle de estado (prancha XI), na linha -IMPEDE, substituir o NOT 5F(5,6) por NAND, com o sinal -INDT na outra entrada.

2º - Existe a possibilidade de um "loop" de indiretos sem fim. Isso pode ser evitado com a inclusão de um contador para contar o número de FASES 6 que ocorrem. Esse contador é limpo pelos sinais LAL ou INDT. Quando esse contador atingir um certo número, desliga-se o flip flop RODA1 e para-se a máquina.

## 7.5 ESTATÍSTICA DAS INSTRUÇÕES

Esta não é uma mudança no sistema, mas a sua utilização para se caracterizar a importância de cada instrução, dependendo do tipo de aplicação.

Em um dos raros artigos que trata da escolha do repertório de instruções de um computador, Buchholz[51] afirma: "na prática, o projetista... inicia com um conjunto (de instruções)baseado na experiência com computadores anteriores. O processo é complexo, envolvendo um gran de trabalho de tentativas e mudanças".

É importante portanto que se possa avaliar a importância de cada instrução, dada as aplicações, para, nos projetos seguintes, dispor-se dessas informações. É possível pensar-se num circuito, ligado a um dos canais de entrada e saída que serve para contar o número de vezes que determinada instrução, especificada pelo programador, é executada.

Na figura 7.6 vê-se um esquema em blocos de um circuito com essa finalidade. Ali se encontra apenas a parte principal, os detalhes foram omitidos. Observe-se ali, que distingue-se as instruções pelos 8 bits do RI, isto é, por exemplo, não se separa os tipos diferentes de deslocamentos cujas indicações se encontram no RD.

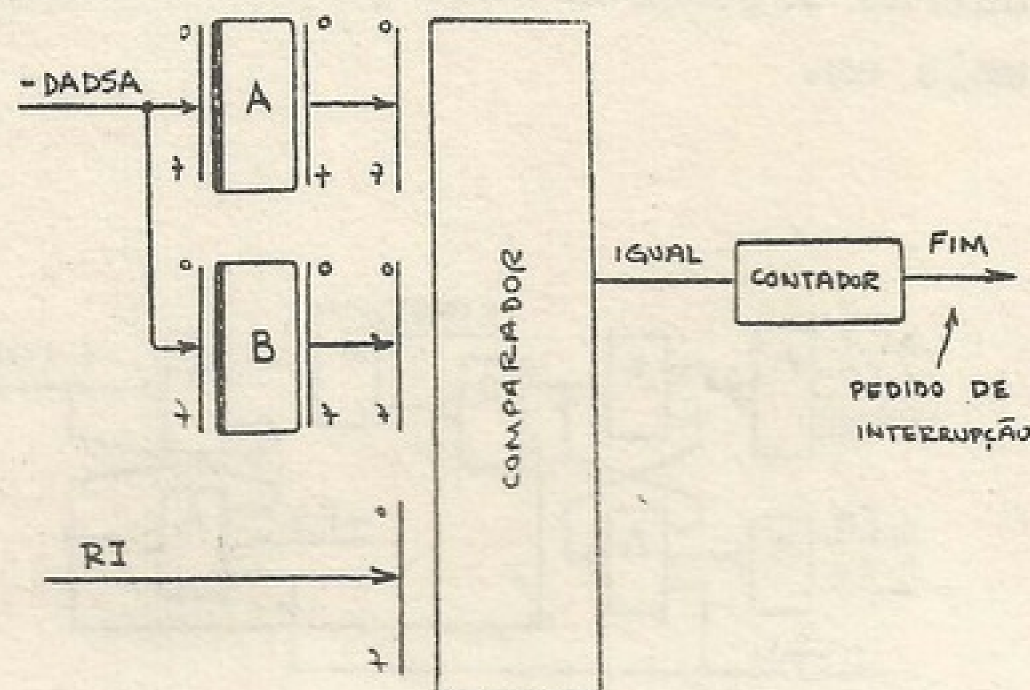



Figura 7.6: Esquema do contador das instruções

A idéia é simples, com uma instrução de saída de dados pelo canal utilizado, o programador carrega o código da instrução que ele quer testar no registrador A. A seguir carrega o complemento desse código no registrador B (ele pode escolher o registrador pelo campo COMANDO da instrução de saída de dados). É essencial que se faça igual a zero os bits irrelevantes ( nos dois registradores) do código. Diferentemente dos outros canais de entrada e saída, este deverá receber os 8 bits do RI. Um circuito comparador (a figura 7.7 mostra um bit desse circuito) dará saída igual a "1" quando o conteúdo do RI for igual ao conteúdo do registrador A (exceptuando os bits irrelevantes), o que abrirá uma porta e no tempo T4 da FASE1 passará um pulso que incrementará de 1 o contador. Quando o contador der "estouro", gerará um pedido de interrupção para uma rotina que incrementará de 1 um contador na memória do computador.

Convém acrescentar-se algumas observações. O esquema indicado funciona contando um tipo de instruções por vez. Mas, pode-se ver que apenas duplicando o esquema, ter-se-á a possibilidade de contar duas instruções diferentes. Como o campo de comando das instruções de entrada e saída tem 4 bits, pode-se pensar em contar até 16 instruções, desde que se multiplique por 16 os circuitos e utilize-se as instruções tipo função para definir qual registrador que se quer carregar, A ou B.

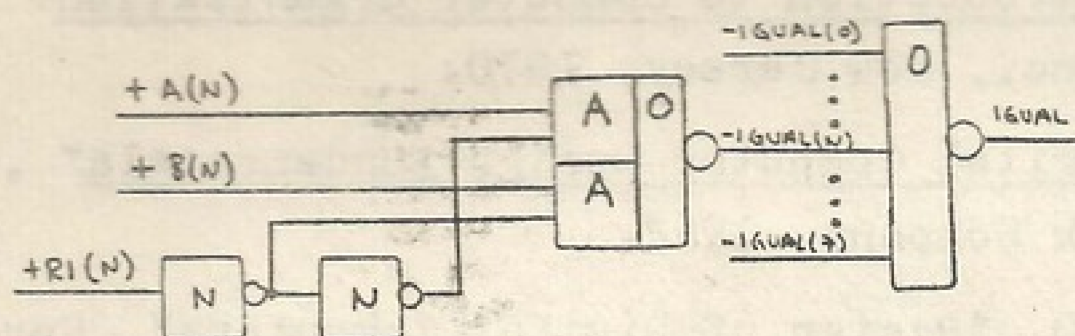



Figura 7.7: Um bit do circuito comparador

Esse esquema cria problemas nas rotinas de entrada e saída. Se o estouro ocorrer quando o sistema estiver atendendo a um pedido de interrupção, o canal contador não conseguirá interromper para ser atualizado. Pode-se pensar, nessas circunstâncias, em se utilizar das instruções de entrada de dados para se carregar, no acumulador, os 8 bits menos significativos do contador, para que se possa saber quantas vezes mais (até 256 no máximo) essa instrução foi executada, após o estouro.

# REFERÊNCIAS


(1) F.J.M. LAVER, "Introducing Computers", London: Her Majesty's Stationery Office, 1965.

(2) J. NIEVERGELT, "Computers and Computing - Past, Present and Future", IEEE Spectrum, janeiro/68, páginas 57 a 61.

(3) WERNER BUCHHOLZ, "Planning a Digital System - Project Stretch", McGraw Hill Book Company, 1962.

(4) ROBERT L. DAVIS, "The Illiac IV Processing Element", IEEE Transactions on Computers, Vol. C-18, Nº 9, Set.69 pg. 800-816.

(5) GLEN GEORGE LANGDON, JR, "Logic Design: A Review and Practice, Part 1: Switching Circuit Technology and Related Timing", SDD Technical Report - IBM Endicott Laboratory, TR01.1370, 1971.

(6) ROBERT F. ROSIN, "Supervisory and Monitor Systems", Computing Surveys, Vol. 1, Nº 1, March/69, pg. 37-54.

(7) DIGITAL EQUIPMENT CORPORATION, "Small Computer Handbook" 1969.

(8) ABHAY K. BHUSHAN, "Guidelines for Minicomputer Selection" Computer Design, Vol. 10, Nº 4, abril/71, pg. 43-57.

(9) J.J. MORRIS, "What to Expect When You Scale Down to a Minicomputer", Control Engineering, Vol. 17, Nº 9, Set. 1970, pg. 65-71.

(10) GLEN GEORGE LANGDON, JR, EDSON FREGNI, "Estruturas de Computadores", Apostila do LSD-EPUSP, 1972.

(11) HERBERT HELLERMAN, "Digital Computer System Principles", McGraw Hill Book Company, 1967

(12) YAOHAN CHU, "Introduction to Computer Organization", Prentice-Hall Inc., New Jersey, 1970.

(13) YAOHAN CHU, "Digital Computer Design Fundamentals", McGraw Hill Book Company, 1962.

(14) HANS W. GSCHWIND, "Design of Digital Computers", Springer-Verlag, Wien, 1967.

(15) HEWLETT-PACKARD, "A Pocket Guide to Hewlett-Packard Computers (2100 Series)", California, 1968.

(16) PAUL SIEGEL, "Understanding Digital Computers", John Wiley and Sons, Inc., New York, 1967.

(17) DONALD EADIE, "Introduction to the Basic Computer" , Prentice Hall Series in Electronic Technology, 1968.

(18) WILLIAM ROBERTS, "Microprogramming Concepts and Advantages as Applied to Small Digital Computers", Computer Design, Vol. 8, Nº 11, Nov./69, pg. 147-150.

(19) WILLIS H. WARE, "Digital Computer Technology and Design", Vol. II (Circuits and Machine Design), John Wiley and Sons, 1966.

(20) SAMIR S. HUSSON, "Microprogramming - Principles and Practices", Prentice-Hall, Inc., N.J., 1970.

(21) P.M. DAVIES, "Readings in Microprogramming", IBM - Systems Journal, Vol. II, Nº 1, 1972, pg. 17-40.

(22) STEPHEN R. REDFIELD, "A Study in Microprogrammed Processors: A Medium Sized Microprogrammed Processor" , IEEE Transactions on Computers, V. C-20, Nº 7, Julho 1971, pg. 743-750.

(23) ROBERT F. ROSIN, "Contemporary Concepts of Microprogramming and Emulation", Computing Surveys, Vol. 1 , Nº 4, Dez/69, pg. 199-212.

(24) GILBERT VANDLING E D. WALDECKER, " The Microprogram Control Technique For Digital Logic Design", Computer Design, Vol. 8, Nº 8, Agosto/69, pg. 44-51.

(25) MICRODATA, "Microprogramming Handbook", Microdata Corporation, California, 1971.

(26) A.M. JOHNSON, "The Microdiagnostics for the IBM System/360 Model 30", IEEE Transactions on Computer , Vol. C-20, Nº 7, Julho/71, pg. 298-803.

(27) RONALD M. GUFFIN, "Microdiagnostics for the Standard Computer MLP-500 Processor", IEEE Transactions on Computer, Vol. C-20, Nº 7, Julho/71, pg. 803-808.

(28) A. GRASSELLI, "The Design of Program-Modifiable Microprogrammed Control Units", IRE Transactions on Electronic Computers, Junho/62, pg. 336-339.

(29) M.J. FLYNN E R.F. ROSIN, "Microprogramming: An Introduction and a Viewpoint", IEEE Transactions on Computer, Vol. C-20, Nº 7, pg. 727-731, julho 1971.

(30) DOROTHY L. SCHNABEL, "The Design of Processor Control Using a Read-Only Storage", IBM Technical Report, TROO.1318, Agosto/65.

(31) FRANK J. LANGLEY, "Small Computer Design Using Microprogramming and Multifunction LSI Arrays", Computer Design, Vol. 9, Nº 4, Abril/70, pg. 151-157.

(32) M.V. WILKES ET AL, "The Design of the Control Unit of an Electronic Digital Computer", Proceedings of the Institution of Electrical Engineers, Vol. 105, Part B, Nº 20, Março/58, pg. 121-128.

(33) M.V. WILKES, "The Growth of Interest in Microprogramming: A Literature Survey", Computing Surveys, Vol. 1, Nº 3, Setembro/69, pg. 139-145.

(34) IVAN FLORES, "The Logic of Computer Arithmetic", Prentice-Hall, 1963.

(35) PHILIPS, "Core Memory Type FI-21", Philips Product Note, Dezembro/70, Electronic Components and Material Division - Holanda.

(36) LANE S. GARRETT, "Integrated-Circuit Digital Logic Families - Part II - TTL Devices", IEEE Spectrum , Novembro/1970, pg. 63-72.

(37) O.L. MACSORLEY, "High-Speed Arithmetic in Binary Computers", Proceedings of the IRE - Special Issue on Computers, Vol 49, Nº 1, Janeiro/61, pg. 67-91.

(38) THOMAS P. SIFFERLEN, "Digital Electronics - With Engineering Applications", Prentice-Hall Series in Computer Applications in Electrical Engineering, N. J., 1970

(39) G. BELL ET AL, "A New Architecture for Minicomputers - The Dec PDP-11", AFIPS - Conference Proceedings , Vol. 36, Spring Joint Computer Conference, 1970, pg. 657-675.

(40) DONALD L. MOON, "Digital Machine Design and Analysis" Computer Design, Julho/70, pg. 59-65.

(41) GLEN GEORGE LANGDON, JR, "Some Definitions and Practices in Logic Design", IBM - SDD Technical Record, TD 01.497, Endicott, 1969.

(42) PAUL W. CASE, "Evolution of Design Automation", Computer (IEEE), Maio-Junho/1972, Vol. 5, Nº 3, pg. 21-35.

(43) GLEN GEORGE LANGDON, JR, "A Survey of Counter Design Techniques", Computer Design, Outubro/70, Vol. 9, Nº 10, pg. 85-93.

(44) GLEN GEORGE LANGDON, JR, "A Review of Theory and Practice - Part III: Logic Design Practices and Interrelationship", SDD Technical Report -IBM, Endicott Laboratory, TR01-469, Maio/1971.

(45) McCLUSCKEY, E.J., "Introduction to the Theory of Switching Circuits", McGraw Hill, N.Y., 1965.

(46) H.F. NAJJAR, "Partition Techniques for Switching Logic", Electro Technology, Abril/1967.

(47) J. PAUL ROTH, "Diagnosis of Automata Failures: A Calculus and a Method", IBM Journal, Julho/1966.

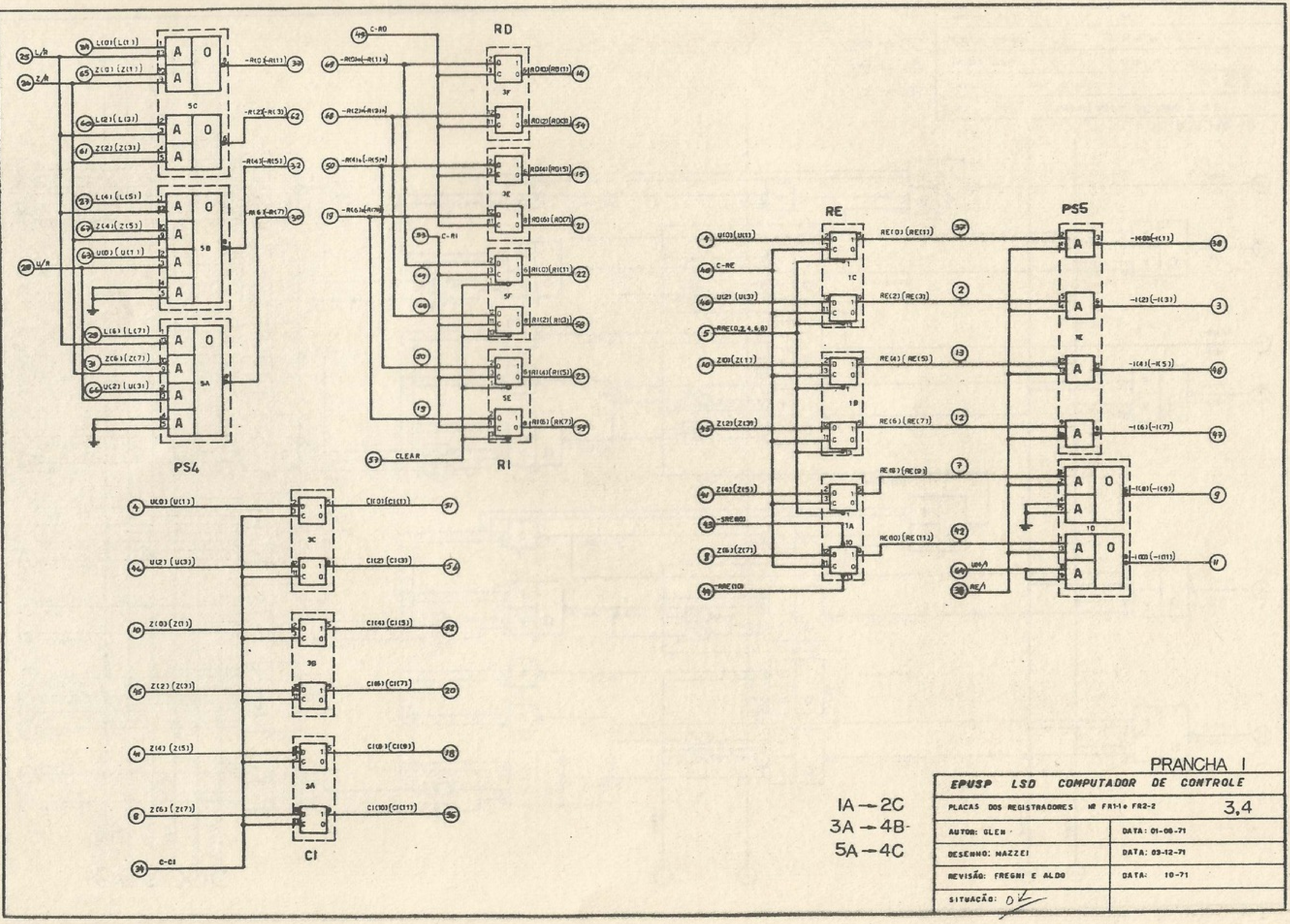
(48) R.J. GIANNUZI, C.S. GURSKI E J.H. STEVENS, "A Computer Aided Packaging and Partitioning System", Proceedings of the Technical Program, California, 1971.

(49) EDSON FREGNI, "Projeto Lógico do Primeiro Computador do LSD", Anais do IV Congresso Nacional de Processamento de Dados (SUCESSU), S.Paulo, Outubro/1971.

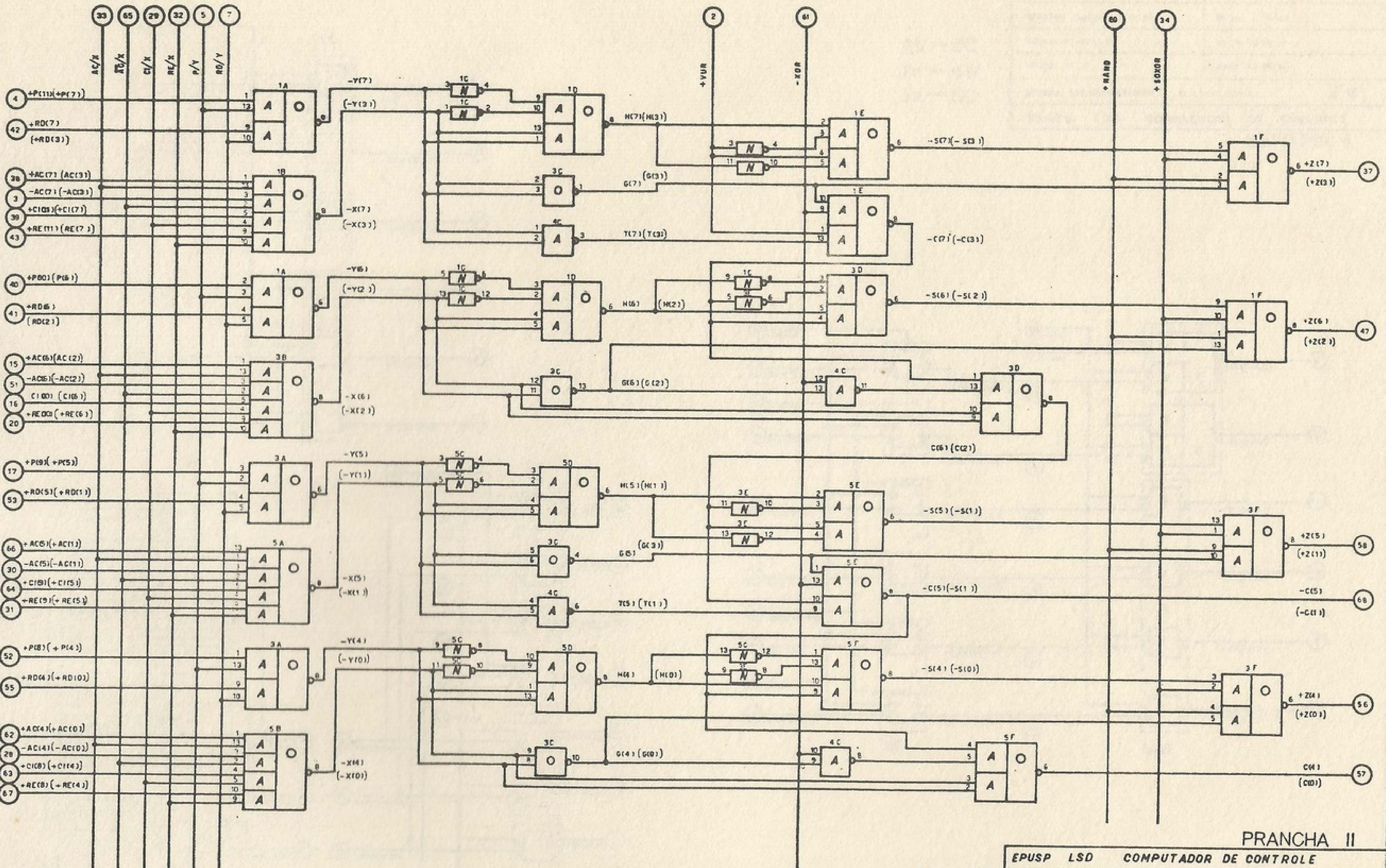
(50) OSCAR LAWENSCHUSS, "Universal LSI Package for Implementing Control Logic Functions", Computer Design , Setembro/1970.

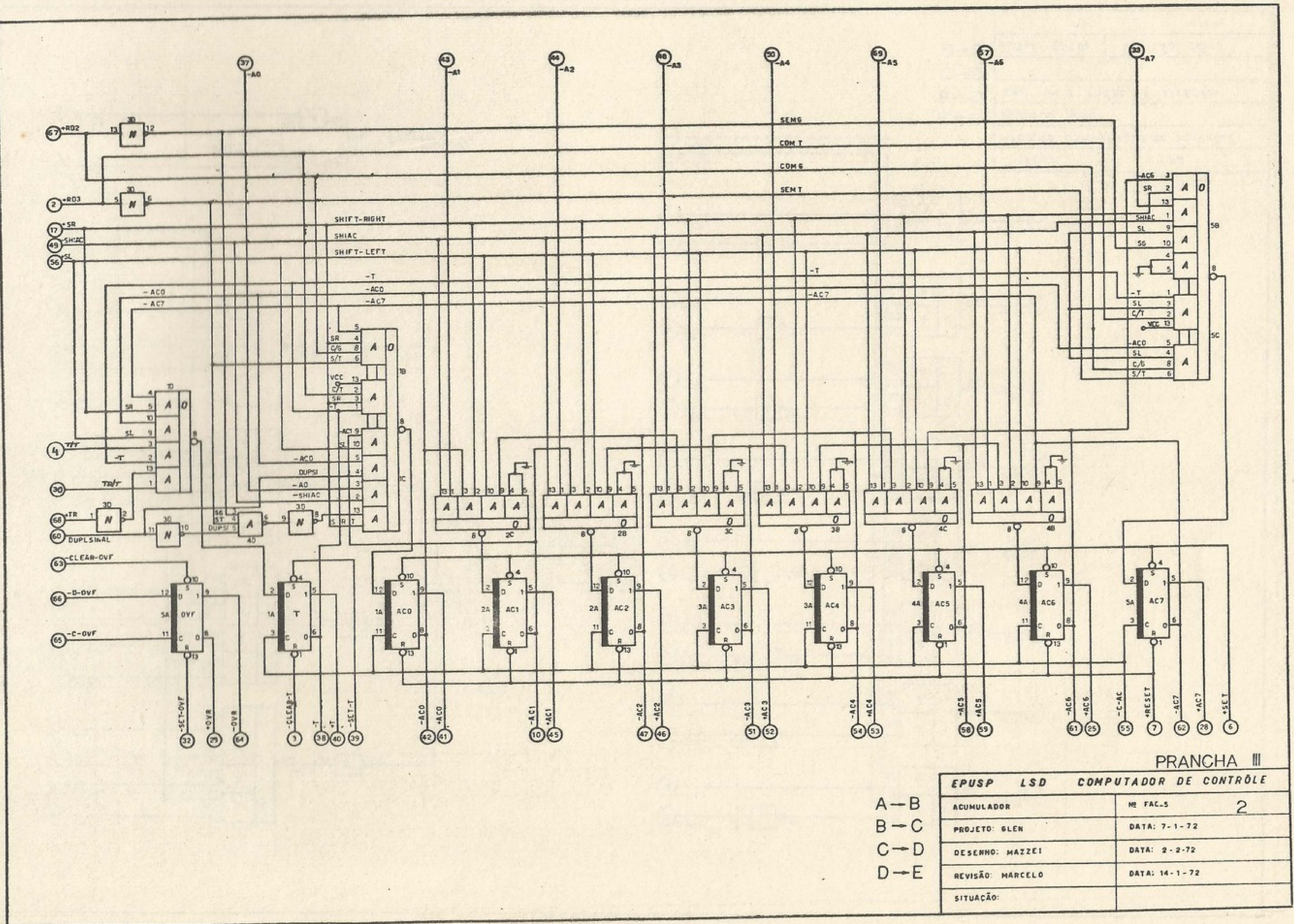
(51) W. BUCHHOLZ, "The Selection of an Instruction Language" , Proceedings of Western Joint Computer Conference, Los Angeles, California, Maio/1958 - Publicado no AIEE Publication, nº T107, março/1959. pg 138-30.

# PRANCHAS

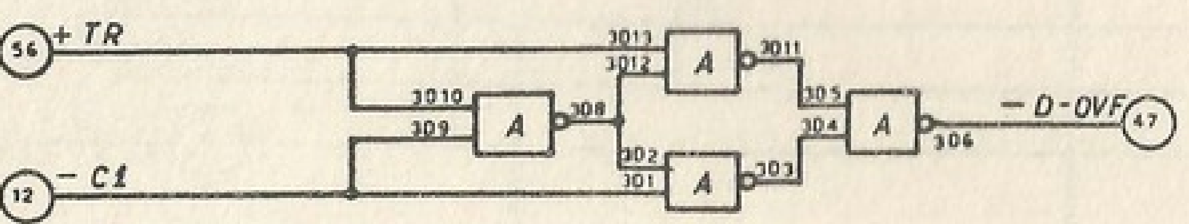


PS4
RD
RI
CI
RE
PS5
CLEAR
C-RD
C-RI
C-RE
C-CI
PRANCHA I
EPUSP LSD COMPUTADOR DE CONTROLE
PLACAS DOS REGISTRADORES Nº FR1-1 e FR2-2
3,4
AUTOR: GLEN
DATA: 01-06-71
DESENHO: MAZZEI
DATA: 03-12-71
REVISÃO: FREGNI E ALDO
DATA: 10-71
SITUAÇÃO: OK
1A → 2C
3A → 4B
5A → 4C


1A → 2B
3A → 4B
5A → 2C
PRANCHA II
EPUSP LSD COMPUTADOR DE CONTROLE
CIRCUITO - UNIDADE ARITMÉTICA
5,6
AUTOR: VICTOR
DATA: 06-07-71
DESENHO: MAZZEI
DATA: 29-11-71
REVISÃO: OK
DATA:



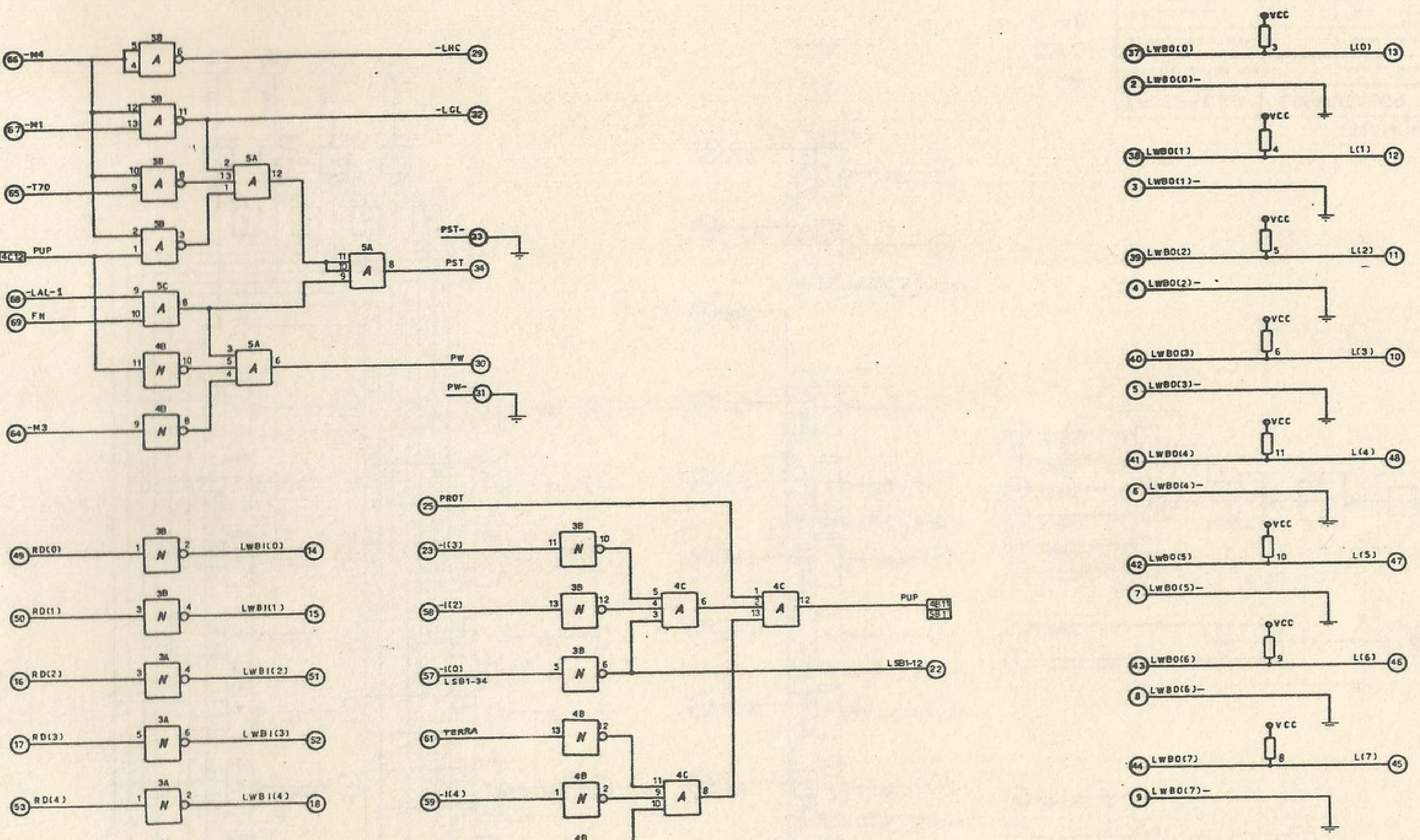
PRANCHA III
EPUSP LSD COMPUTADOR DE CONTRÔLE
ACUMULADOR
Nº FAC-5
2
PROJETO: GLEN
DATA: 7-1-72
DESENHO: MAZZEI
DATA: 2-2-72
REVISÃO: MARCELO
DATA: 14-1-72
SITUAÇÃO:
A→B
B→C
C→D
D→E
SEMG
COM T
COM G
SEM T
SHIFT-RIGHT
SHIAC
SHIFT-LEFT
OVF
T
AC0
AC1
AC2
AC3
AC4
AC5
AC6
AC7
+RD2
+RD3
+SR
SHIAC
SL
DUPLSINAL
-CLEAR-OVF
-D-OVF
-C-OVF
-SET-OVF
+OVR
-OVR
-CLEAR-T
-T
+T
-SET-T
-AC0
+AC0
-AC1
+AC1
-AC2
+AC2
-AC3
+AC3
-AC4
+AC4
-AC5
+AC5
-AC6
+AC6
-C-AC
+RESET
-AC7
+AC7
+SET
-A0
-A1
-A2
-A3
-A4
-A5
-A6
-A7



PRANCHA IV

| | | |
|---|---|---|
| EPUSP | LSD | 7 |
| PROJETO COMPUTADOR DE CONTRÔLE | | |
| FLUXO DE DADOS | | |
| MAIS UM E PORTA DE SELEÇÃO | | |
| AUTOR: EDSON | REVISÃO: CÉLIO | |
| OK | | 20-10-71 |

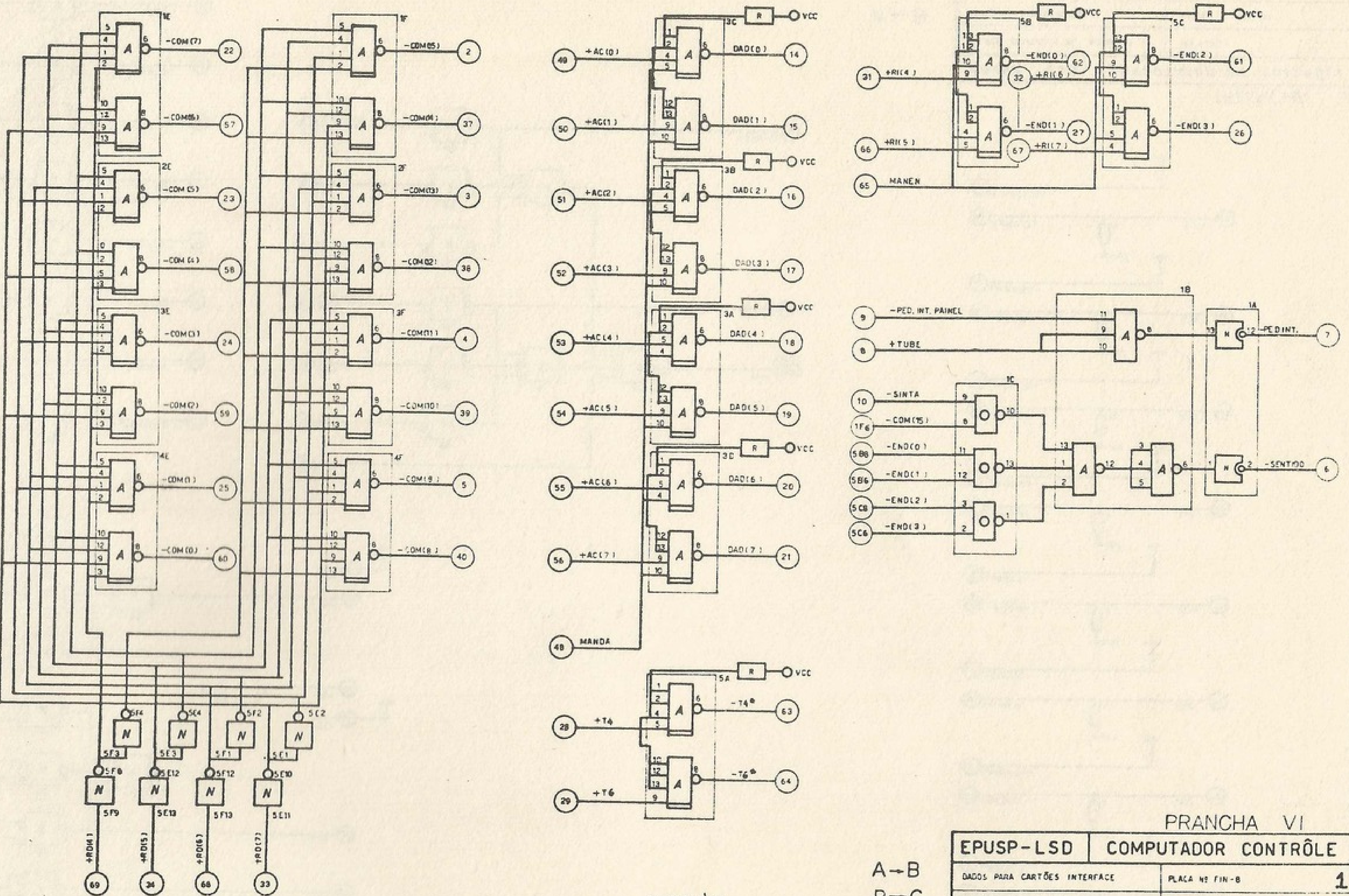
A→B
B→C
C→D
D→E




PRANCHA V

A — B
B — C
C — D

| EPUSP LSD COMPUTADOR DE CONTRÔLE | |
|---|---|
| PLACA CONTRÔLE DA MEMÓRIA | Nº FCM_7 |
| PROJETO: EDITH/PATULLO | DATA: 8 - 71 |
| DESENHO: MAZZEI | DATA 27 - 12 - 71 |
| REVISÃO: ROGER | DATA: 3 - 1 - 72 |
| SITUAÇÃO: | |


PRANCHA VI
EPUSP-LSD
COMPUTADOR CONTRÔLE
DADOS PARA CARTÕES INTERFACE
PLACA Nº FIN-8
1
AUTOR: EDSON FREGNI
DATA: 15-10-71
DESENHO: FREGNI
DATA: 1-11-71
REVISÃO: FREGNI
DATA: 15-11-71
SITUAÇÃO OK
A→B
B→C
C→D
3D→2C
R = 460 Ω . ¼ W
MANDA
MANEN
-PED. INT. PAINEL
+TUBE
-SINTA
-PED INT.
-SENTIDO
VCC

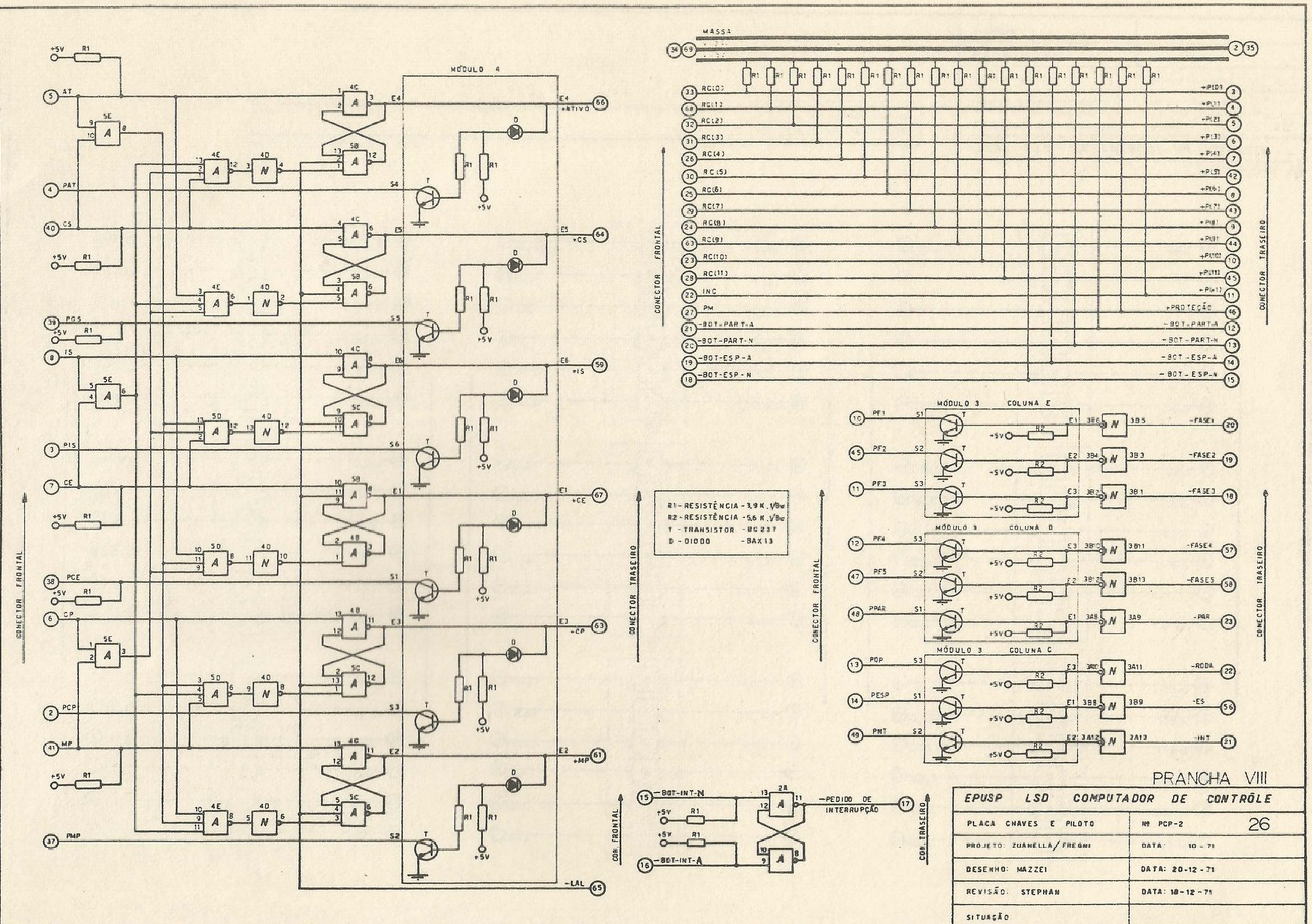
PRANCHA VII

| EPUSP LSD COMPUTADOR DE CONTROLE | |
| --- | --- |
| PLACA MOSTRA REGISTRADORES | Nº PMR-1 25 |
| PROJETO: ZUANELLA/FREGNI | DATA: 10 - 71 |
| DESENHO: MAZZEI | DATA: 7 - 1 - 72 |
| REVISÃO: STEPHAN | DATA: 5 - 1 - 72 |
| SITUAÇÃO: | |

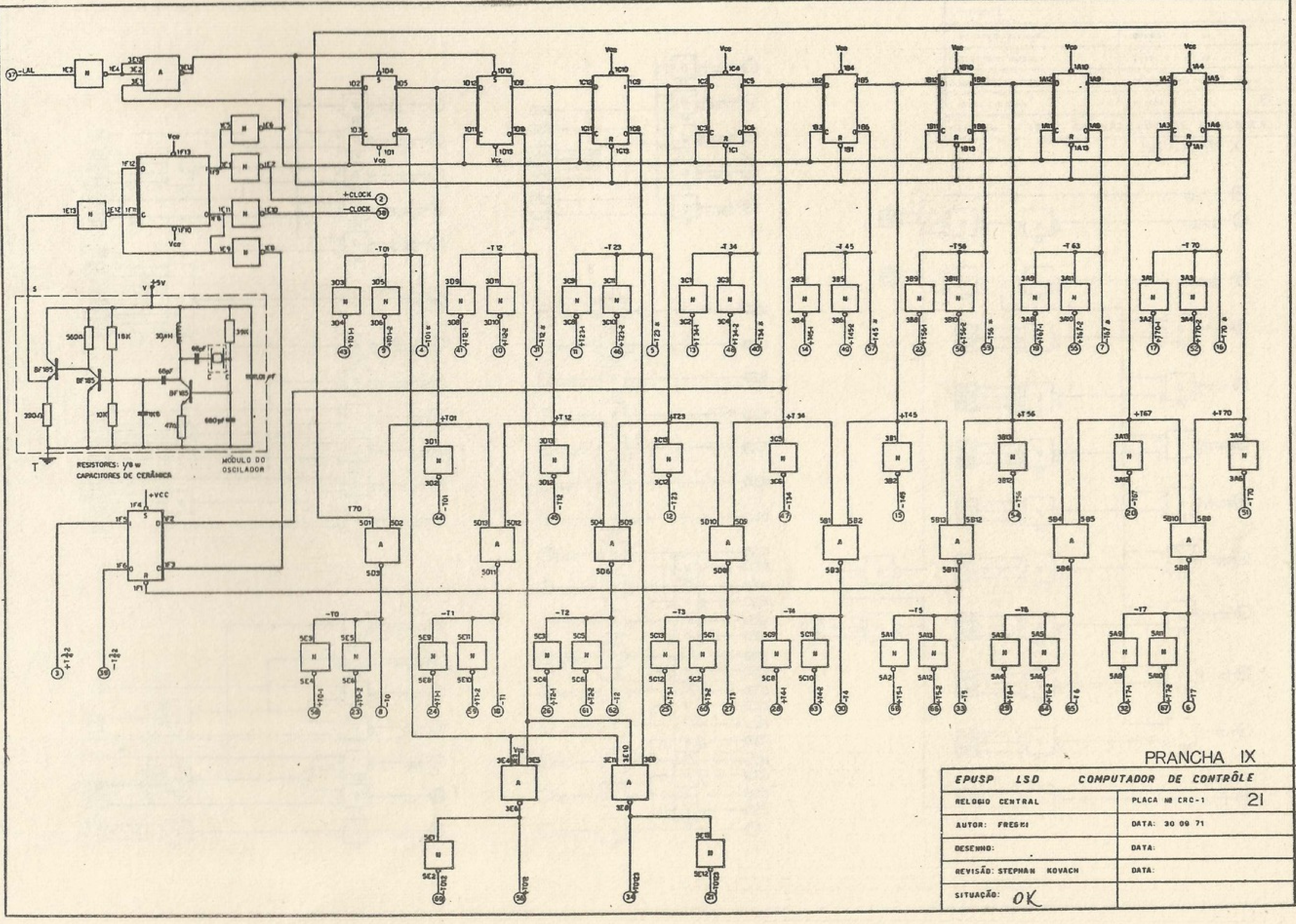
V=5V 36, 42, 53, 64, 70

MASSA 1, 7, 18, 29, 35

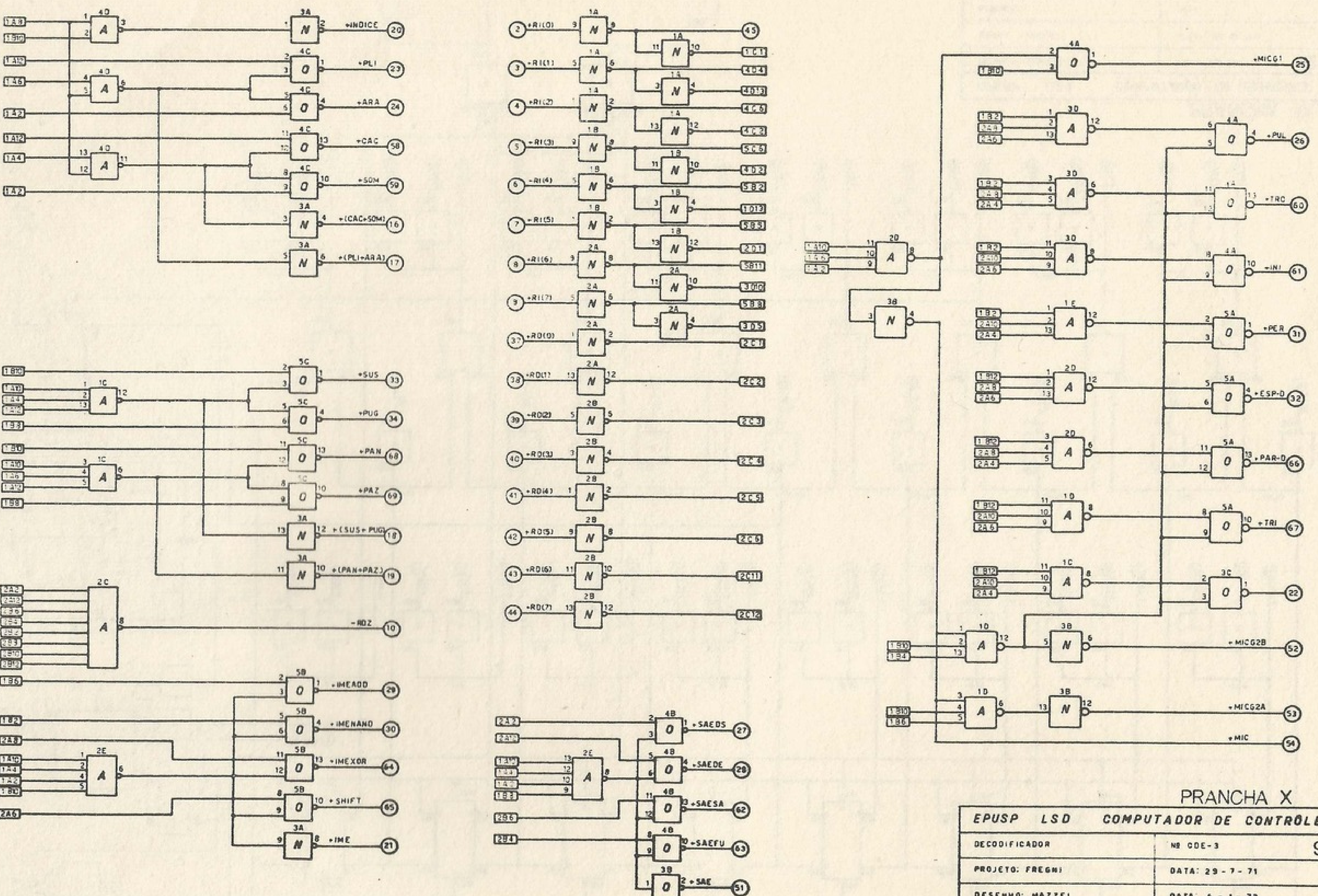
F - CONECTOR FRONTAL
T - CONECTOR TRASEIRO




PRANCHA VIII

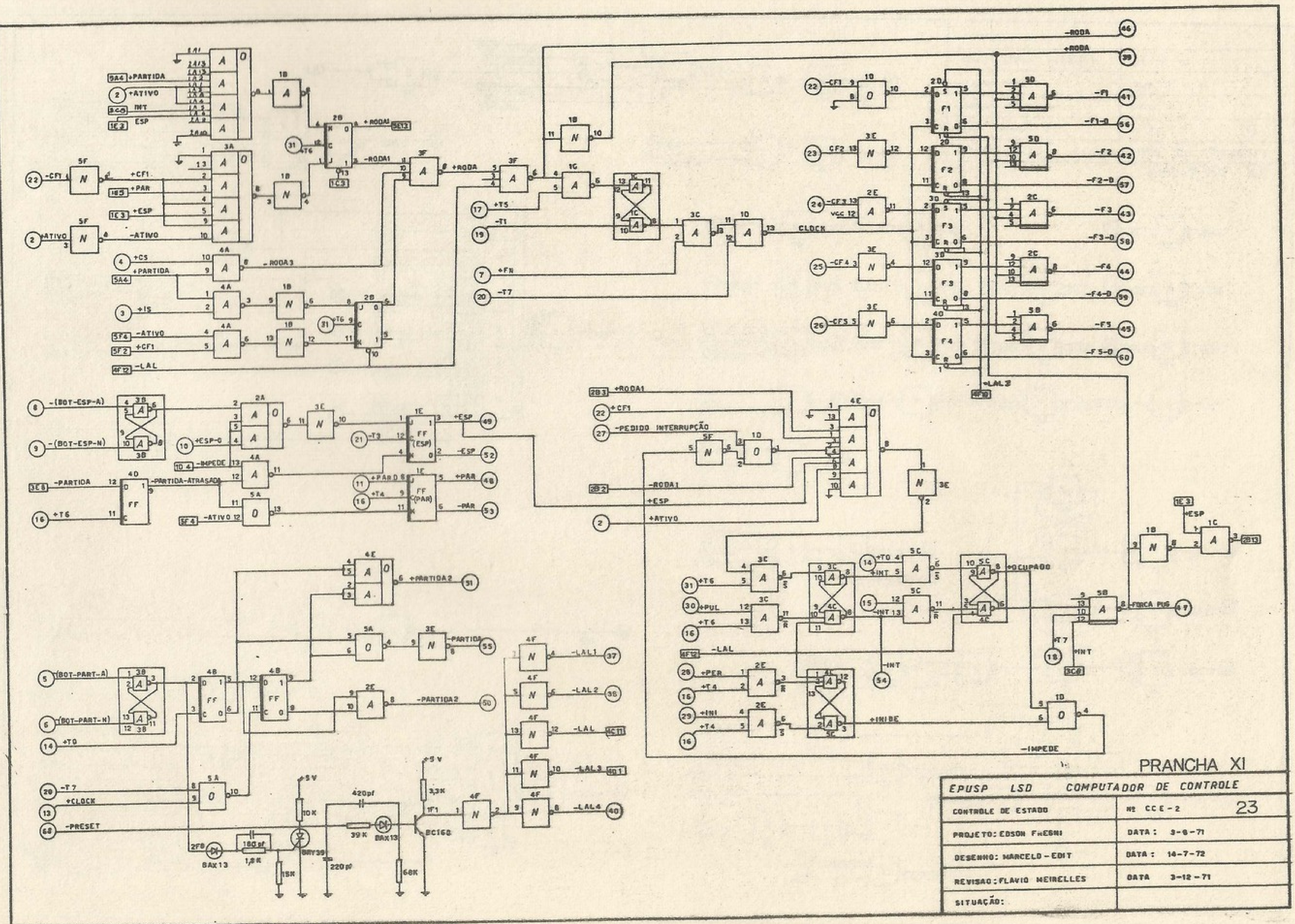


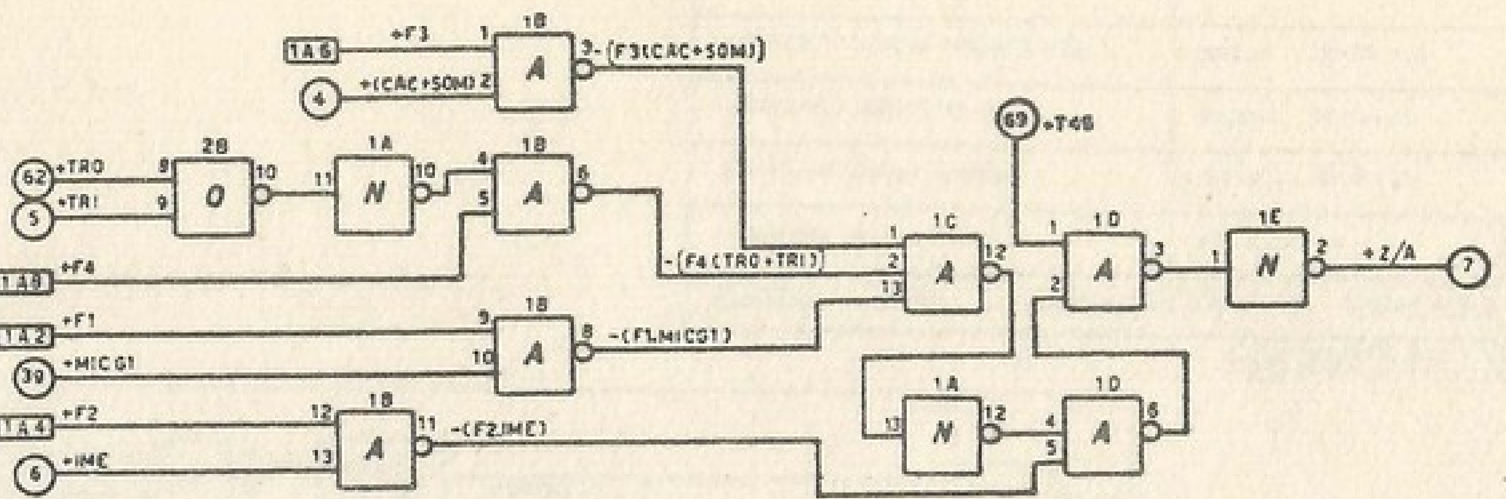
PRANCHA IX
EPUSP LSD COMPUTADOR DE CONTRÔLE
RELOGIO CENTRAL
PLACA Nº CRC-1
21
AUTOR: FREGNI
DATA: 30 09 71
DESENHO:
DATA:
REVISÃO: STEPHAN KOVACH
DATA:
SITUAÇÃO: OK
RESISTORES: 1/8 w
CAPACITORES DE CERÂMICA
MÓDULO DO OSCILADOR
+CLOCK
-CLOCK

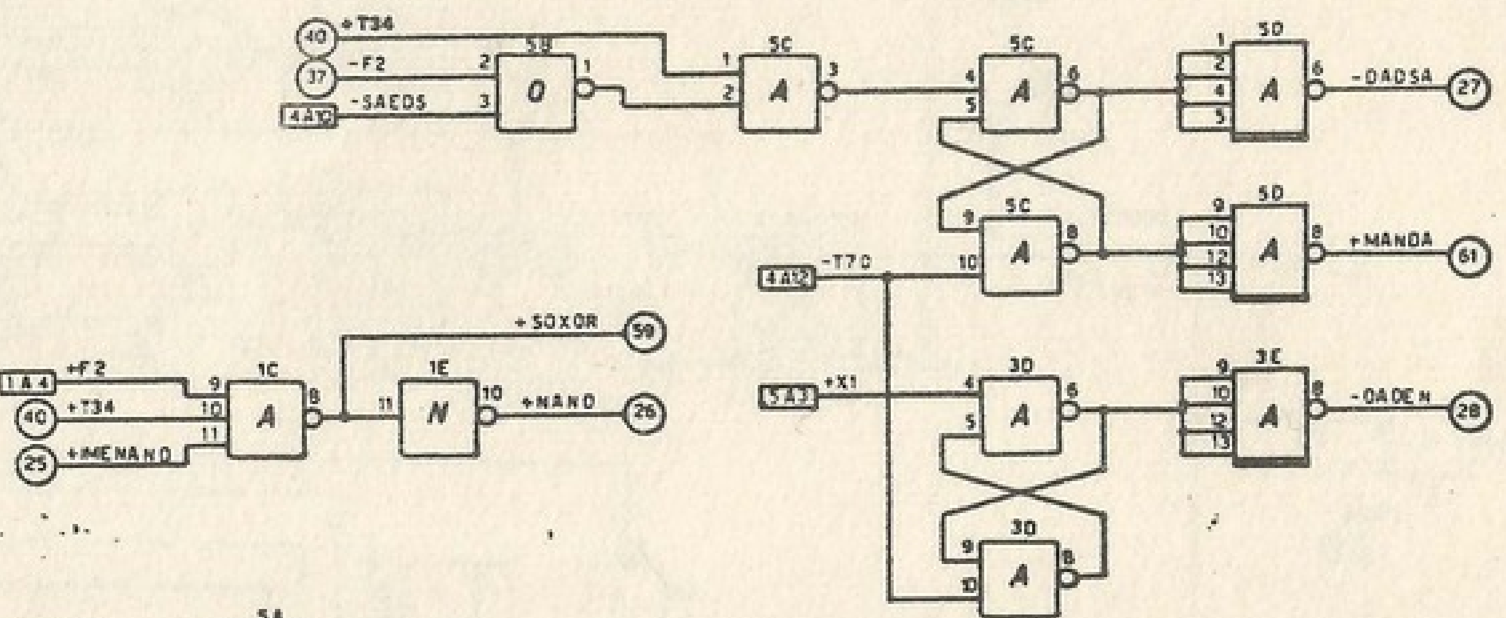


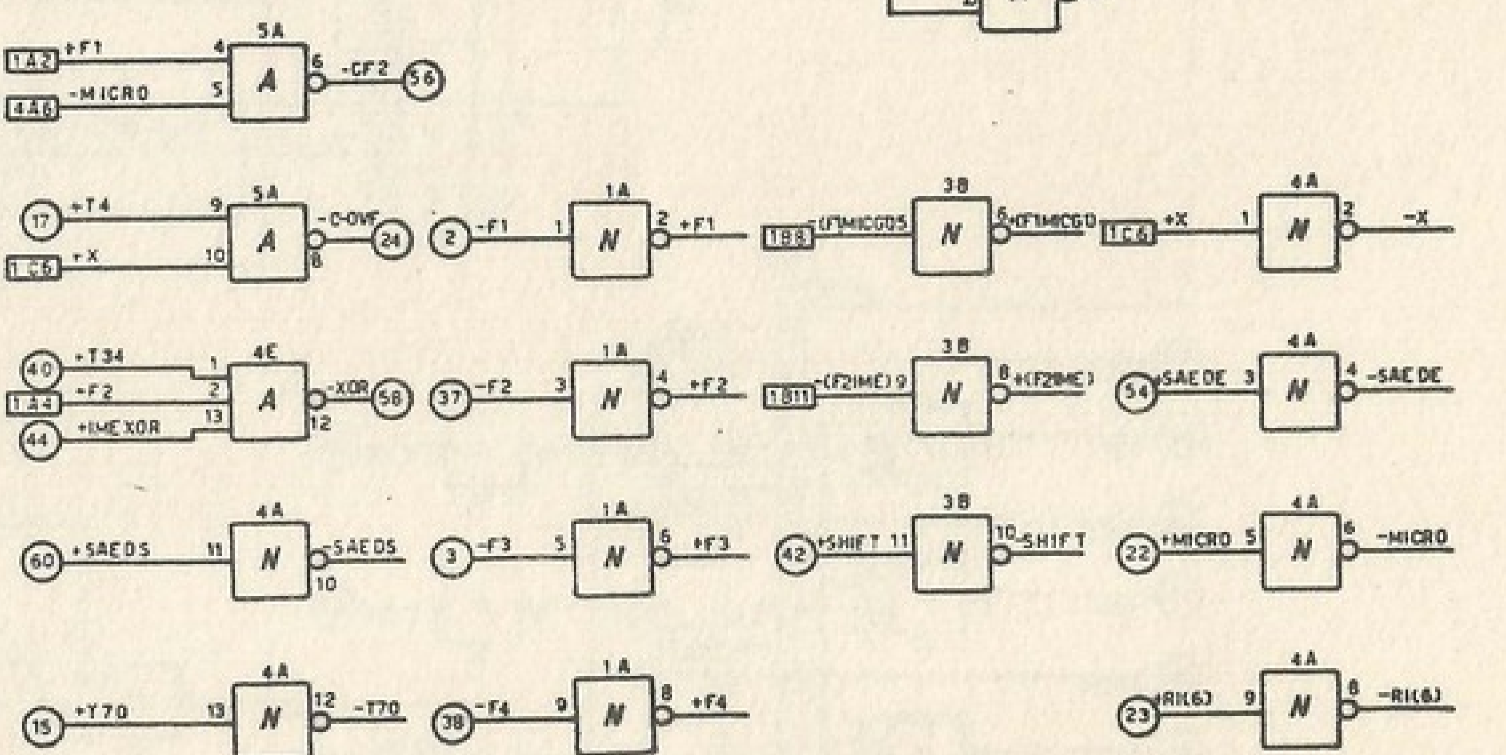
PRANCHA X

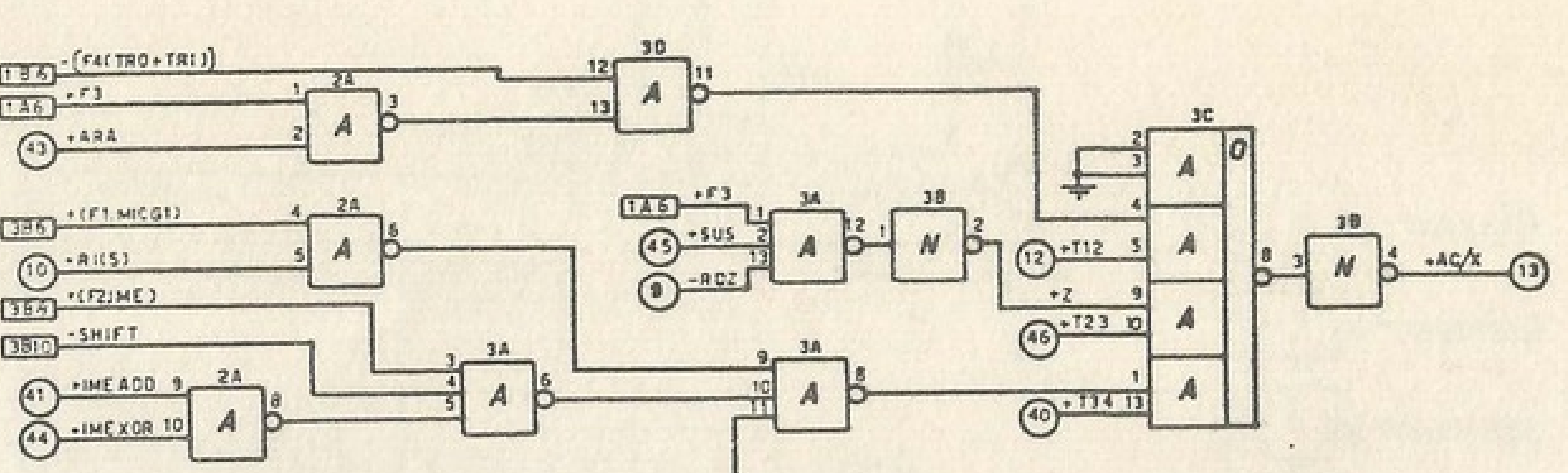
| | |
|---|---|
| EPUSP LSD | COMPUTADOR DE CONTRÔLE |
| DECODIFICADOR | Nº CDE-3 9 |
| PROJETO: FREGNI | DATA: 29 - 7 - 71 |
| DESENHO: MAZZEI | DATA: 6 - 1 - 72 |
| REVISÃO: STEPHAN | DATA: 21 - 12 - 71 |
| SITUAÇÃO: | |

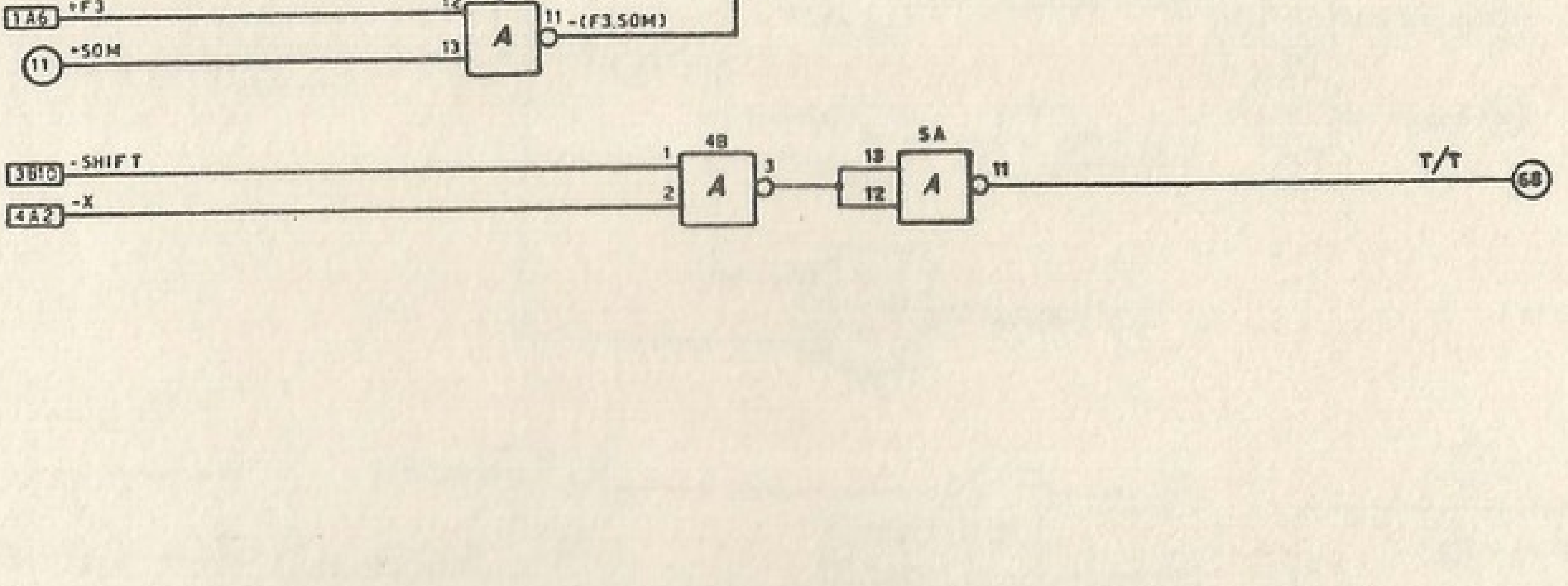

PRANCHA XI
EPUSP LSD COMPUTADOR DE CONTROLE
CONTROLE DE ESTADO
Nº CCE-2
23
PROJETO: EDSON FREGNI
DATA: 3-6-71
DESENHO: MARCELO-EDIT
DATA: 14-7-72
REVISAO: FLAVIO MEIRELLES
DATA 3-12-71
SITUAÇÃO:

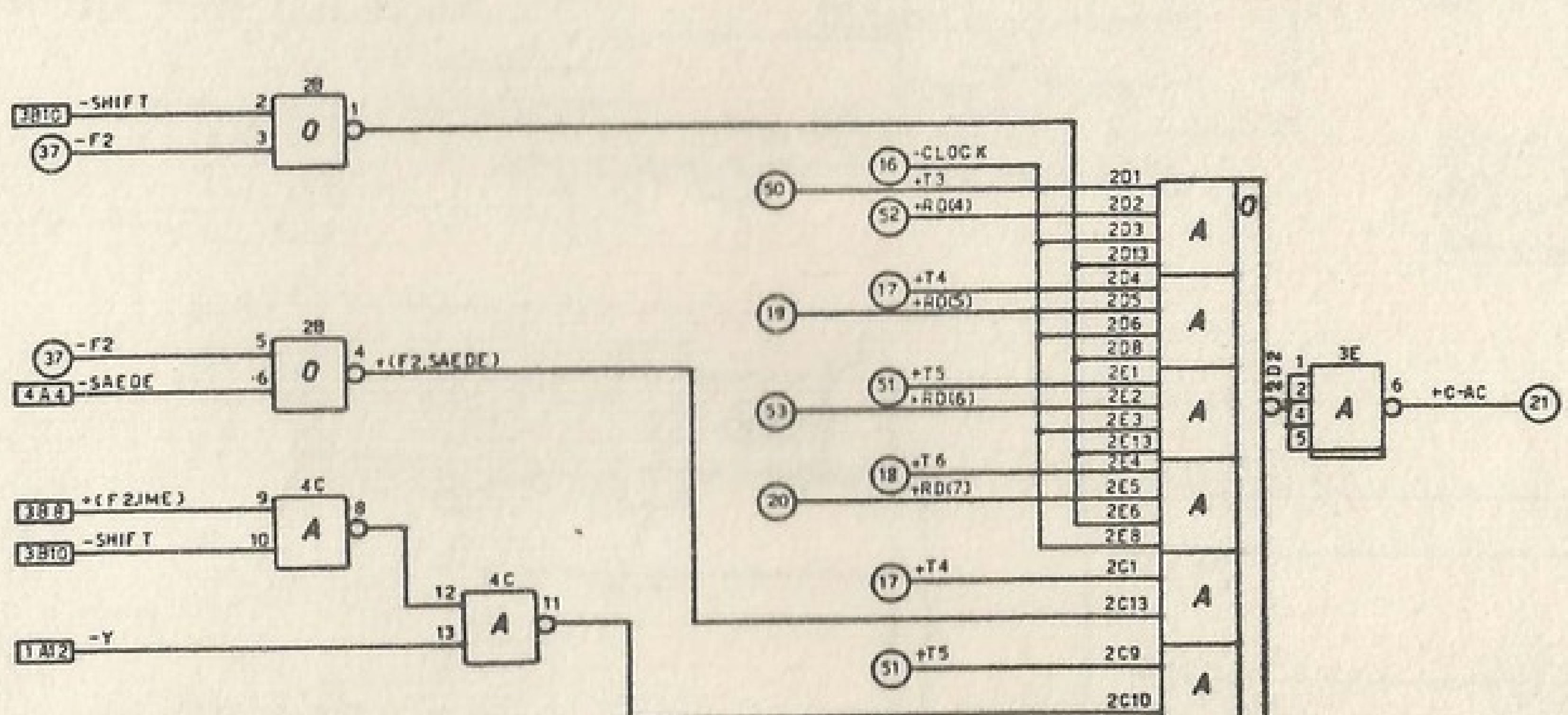


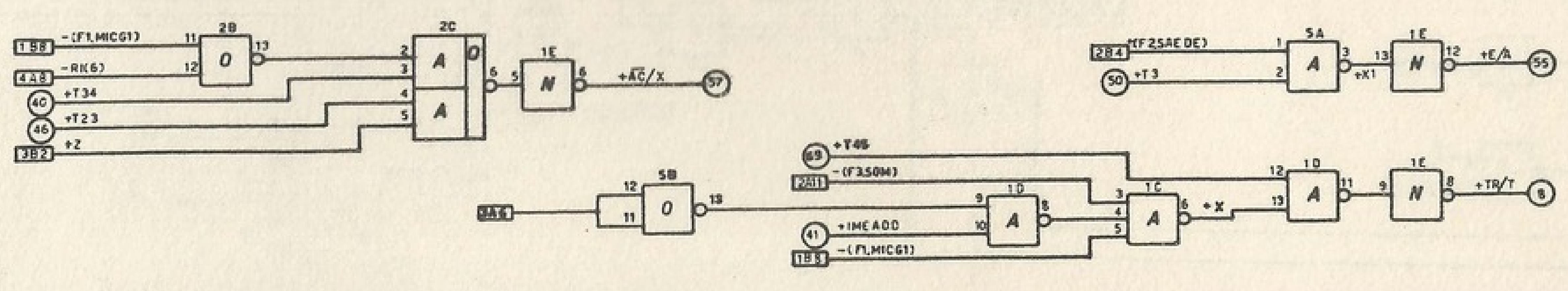






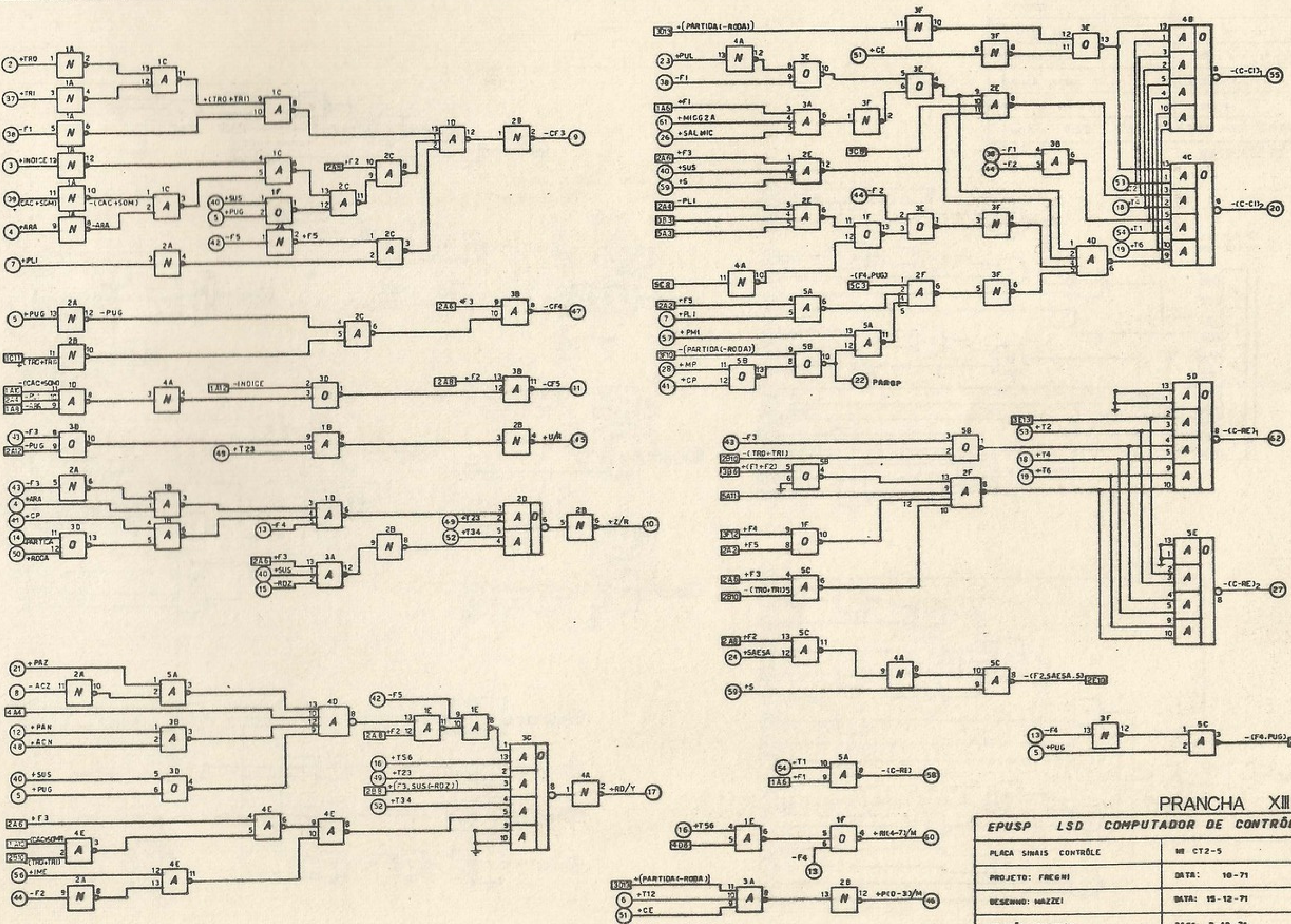




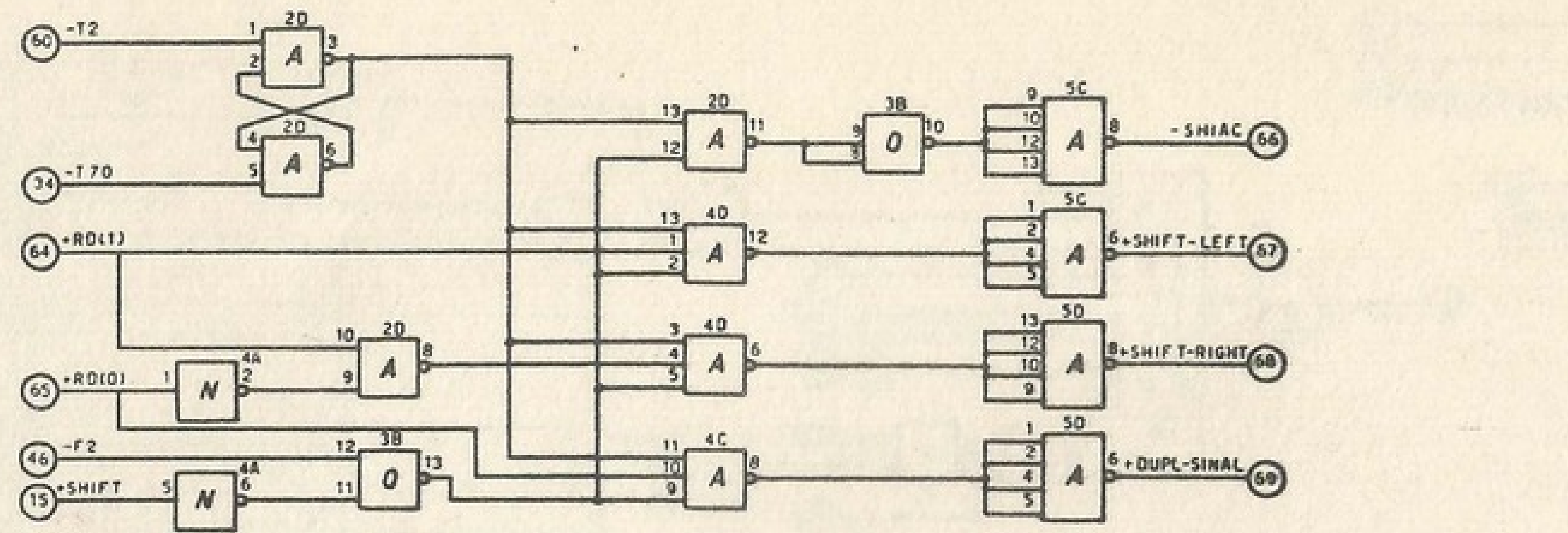


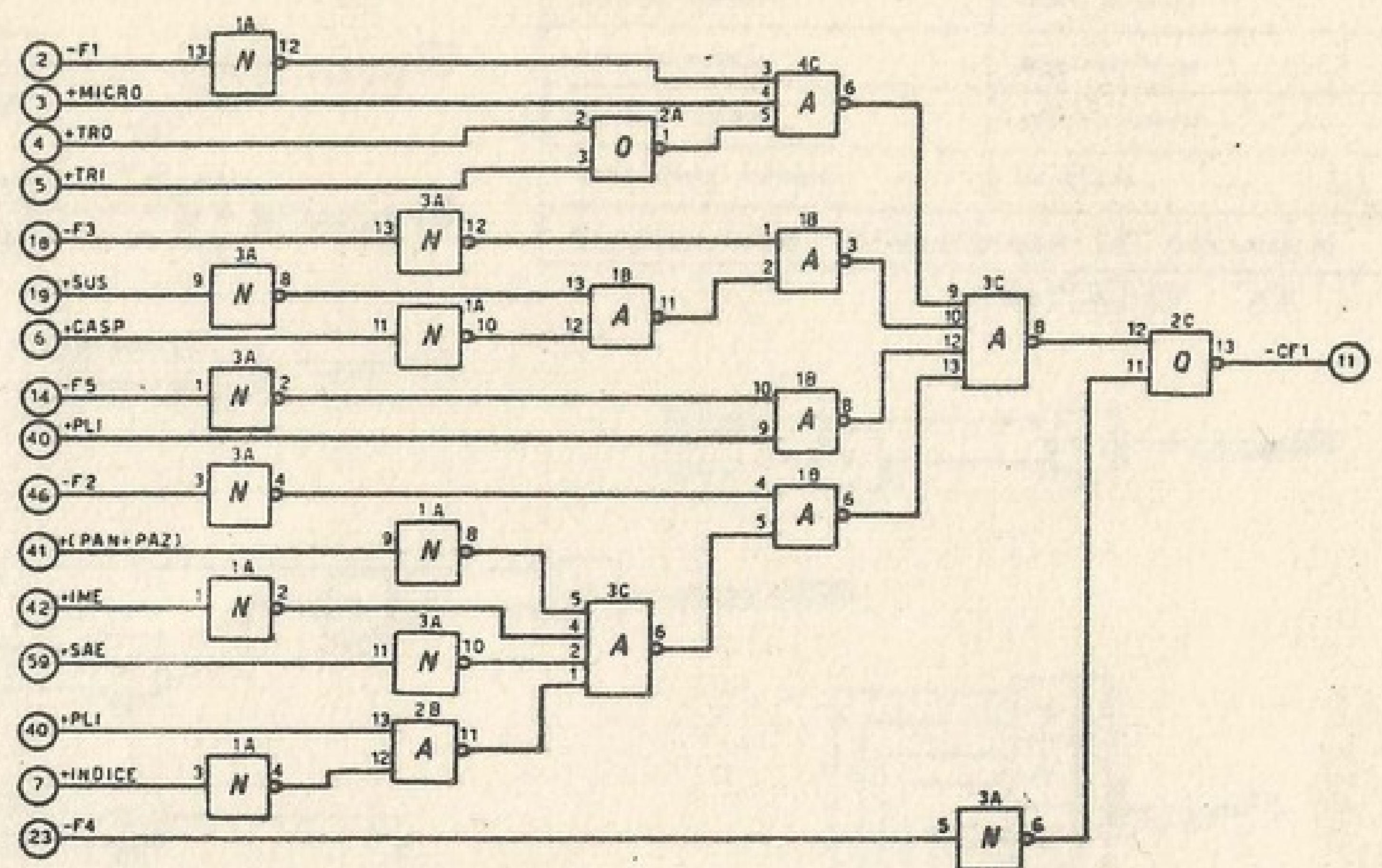
PRANCHA XII

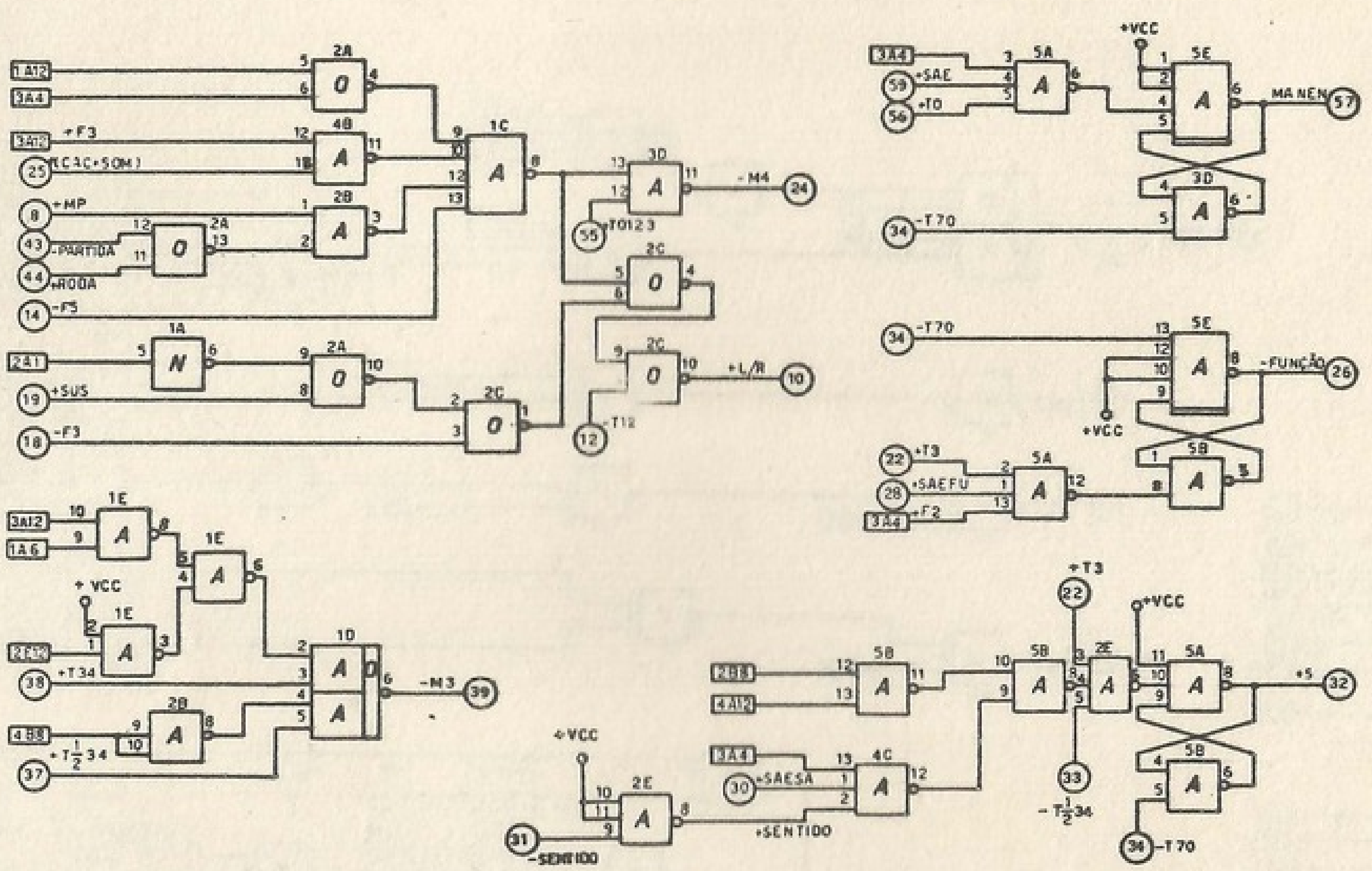
| EPUSP LSD COMPUTADOR DE CONTRÔLE | |
|---|---|
| PLACA CONTRÔLE 1 | Nº CTI - 4 19 |
| PROJETO: FREGNI | DATA: 10 - 71 |
| DESENHO: MAZZEI | DATA: 21 - 12 - 71 |
| REVISÃO: FLÁVIO MEIRELLES | DATA: 15 - 12 - 71 |
| SITUAÇÃO: | |

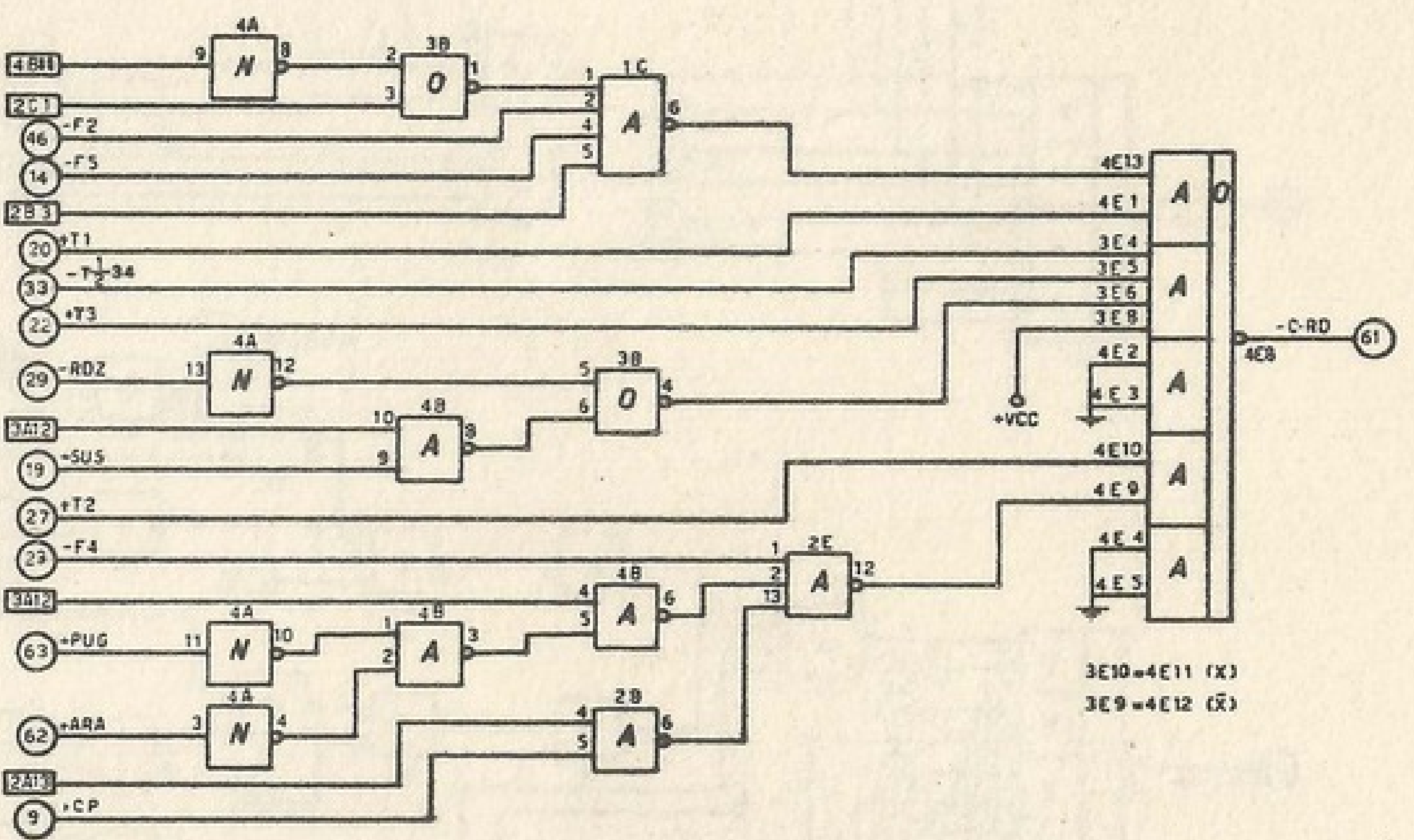


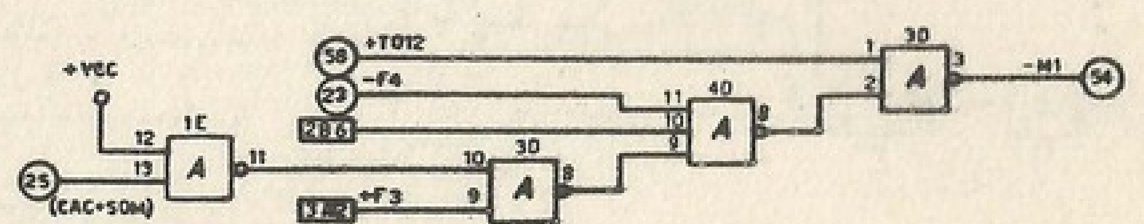
PRANCHA XIII
EPUSP LSD COMPUTADOR DE CONTRÔLE
PLACA SINAIS CONTRÔLE
Nº CT2-5
17
PROJETO: FREGNI
DATA: 10-71
DESENHO: MAZZEI
DATA: 15-12-71
REVISÃO: STEPHAN
DATA: 3-12-71
SITUAÇÃO:



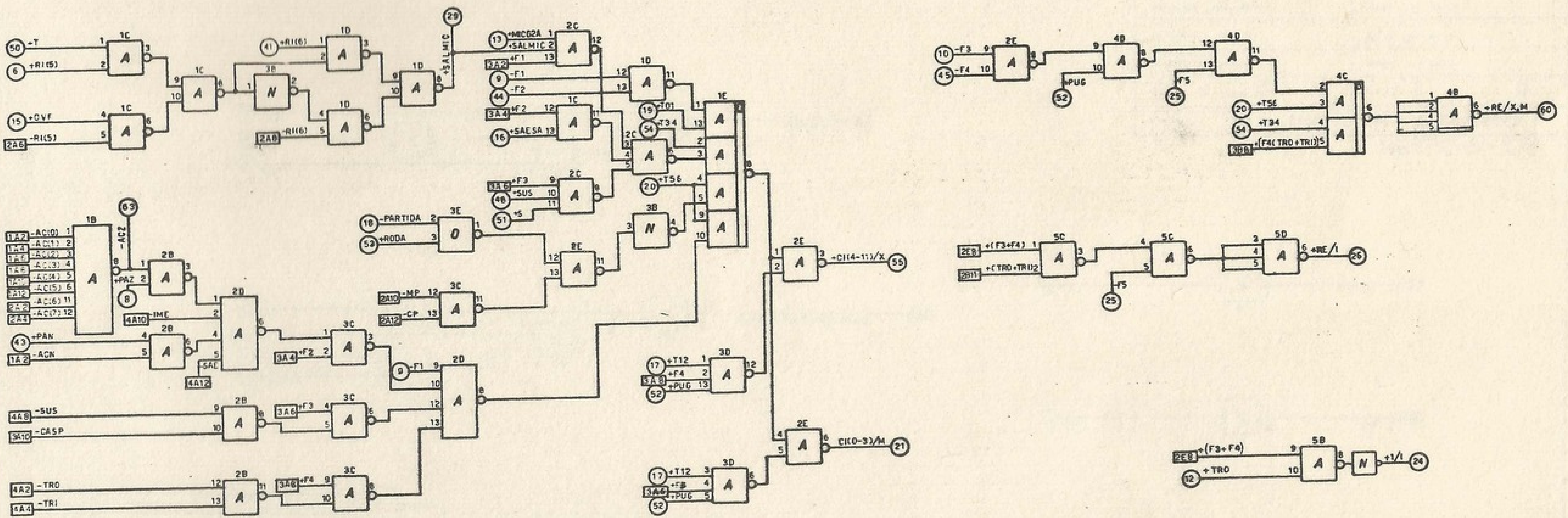



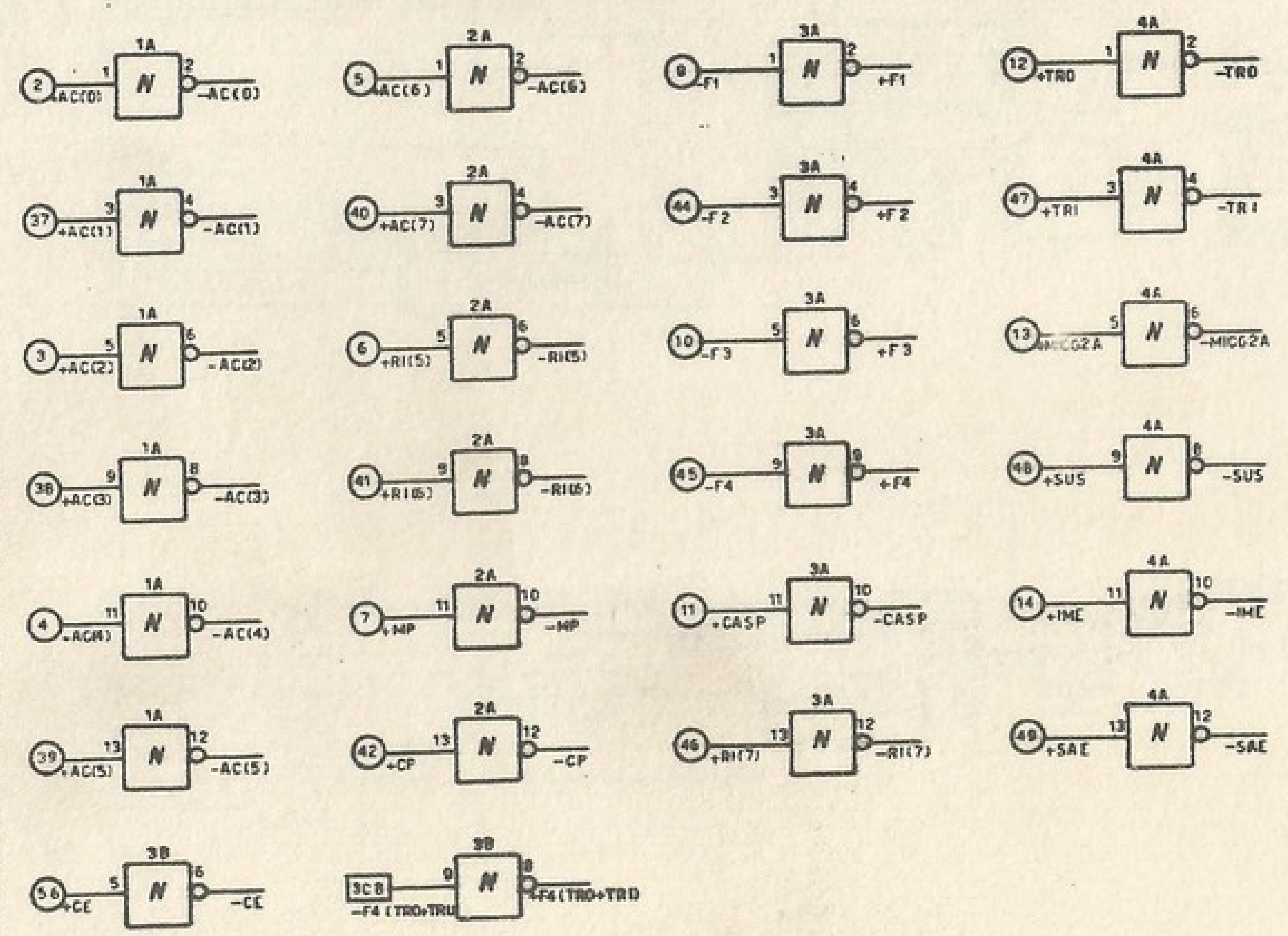



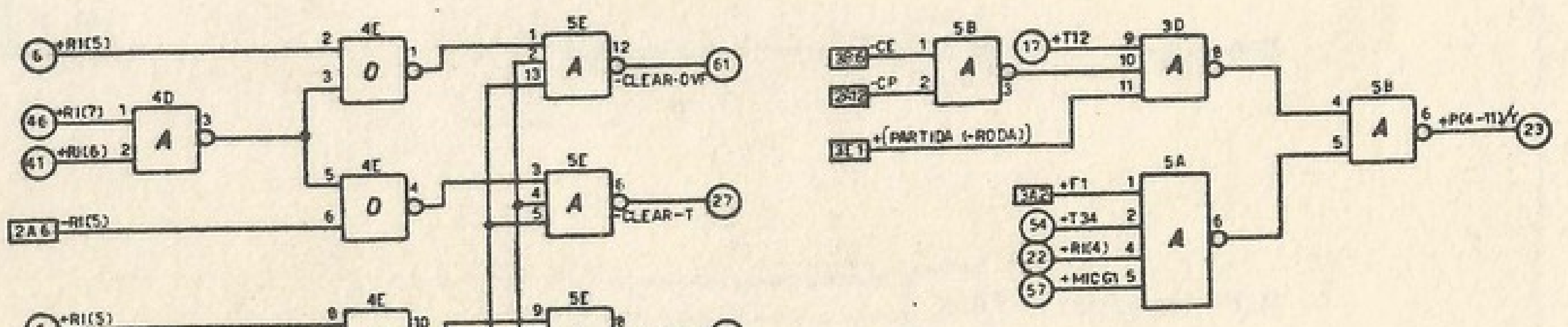




PRANCHA XIV

| EPUSP LSD COMPUTADOR DE CONTRÔLE | |
|---|---|
| PLACA DE CONTRÔLE | Nº CT3-6 15 |
| PROJETO: FREGNI | DATA: 8-71 |
| DESENHO: MAZZEI | DATA: 3-1-72 |
| REVISÃO: EDIT | DATA: 10-11-71 |
| SITUAÇÃO: | |

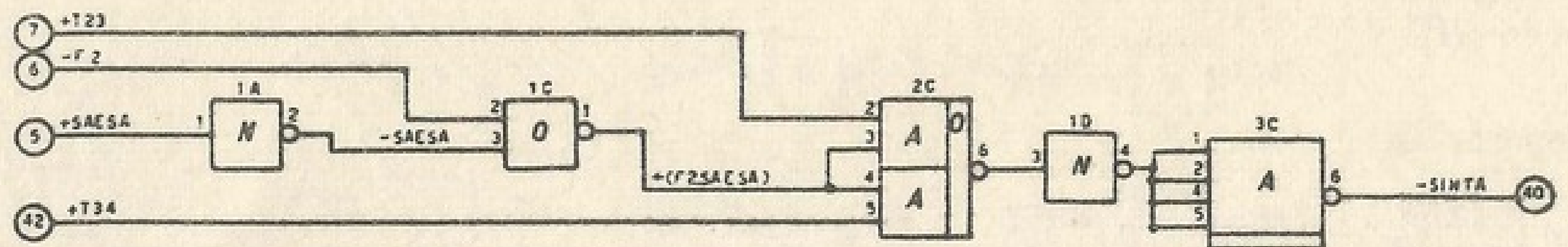


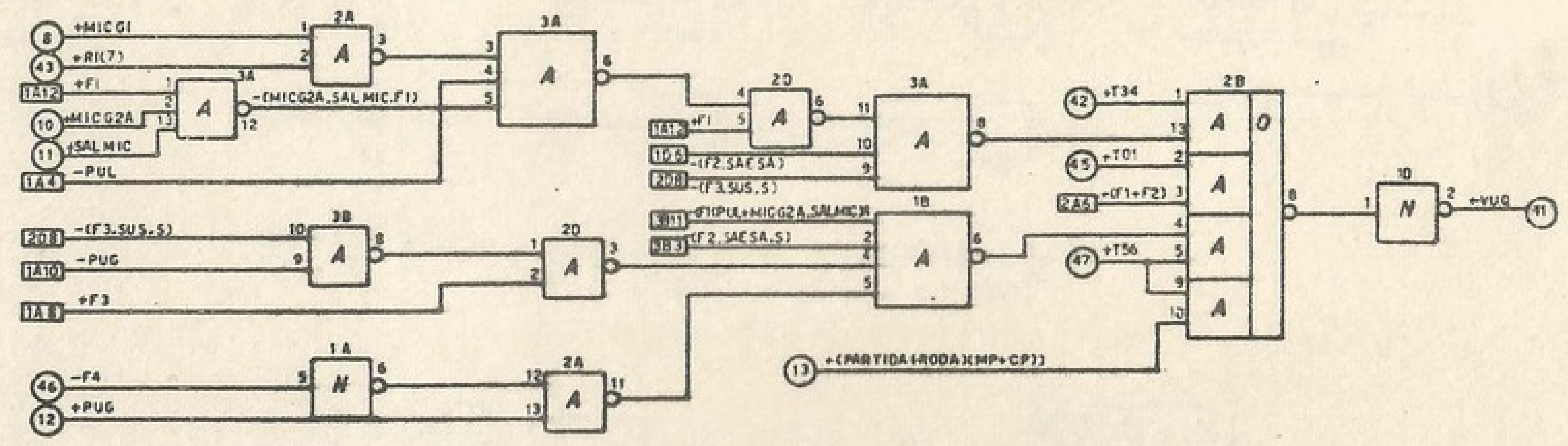


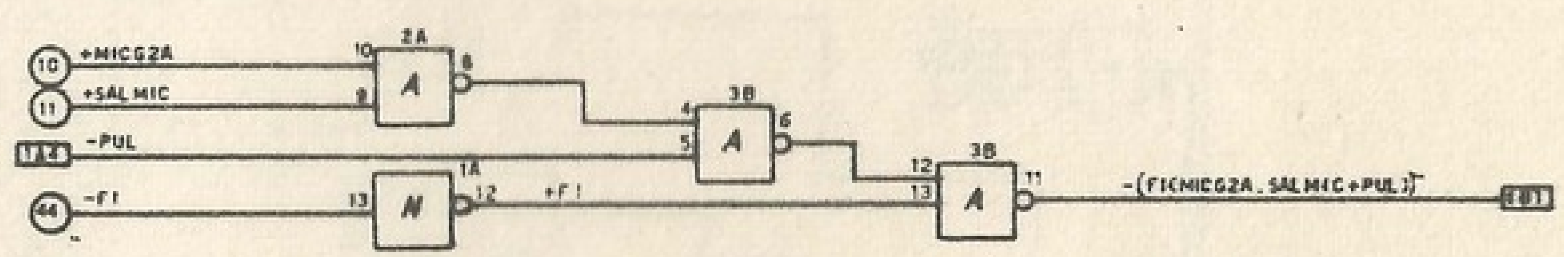


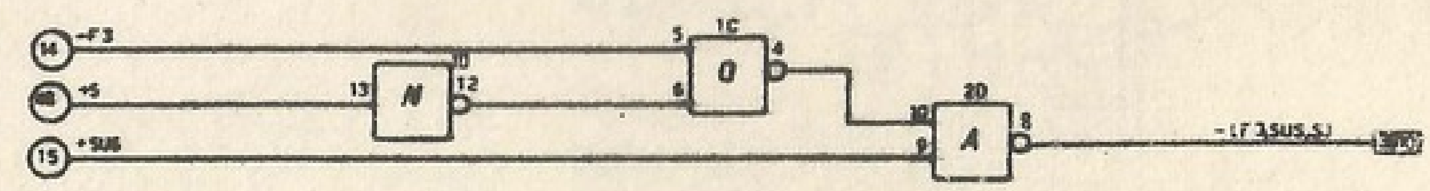
PRANCHA XV

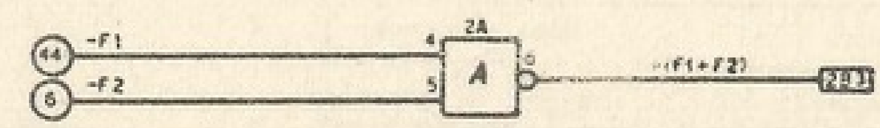
| EPUSP LSD | COMPUTADOR DE CONTRÔLE |
|---|---|
| SINAIS DE CONTRÔLE | Nº CT4-7 13 |
| PROJETO: E. FREGNI | DATA: 10-71 |
| DESENHO: MAZZEI | DATA: 11-12-71 |
| REVISÃO: FLÁVIO MEIRELLES | DATA: 3-12-71 |
| SITUAÇÃO: | |

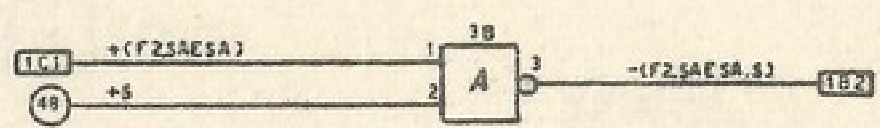


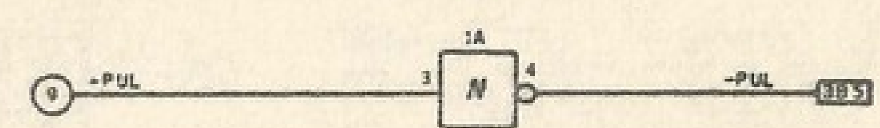



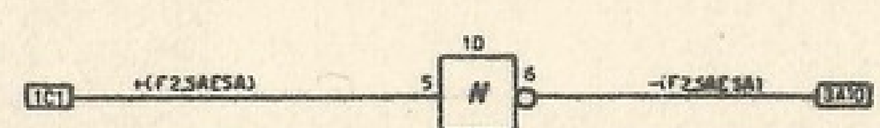



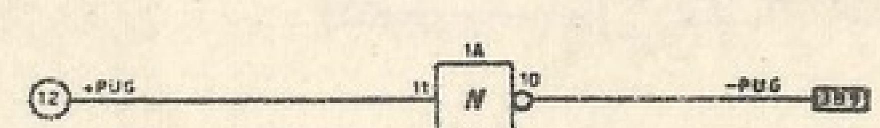



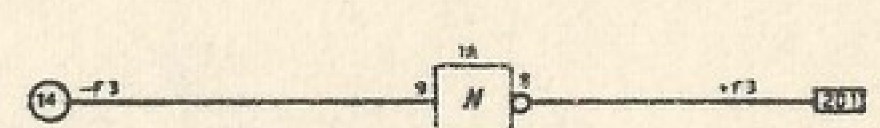




PRANCHA XVI

| EPUSP LSD | COMPUTADOR DE CONTROLE |
|---|---|
| SINAIS DE CONTRÔLE | Nº CTS-4 II |
| PROJETO: FREGNI | DATA: 10-71 |
| DESENHO: MAZZEI | DATA: 16-12-71 |
| REVISÃO: STEPHAN | DATA: 14-12-71 |
| SITUAÇÃO: | |